МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ
УНІВЕРСИТЕТ ІМЕНІ ІВАНА ПУЛЮЯ

Кафедра комп'ютерних систем та мереж

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни
„Комп'ютерна логіка"
(4 семестр)
для студентів денної форми навчання
за спеціальністю
123 „Комп'ютерна інженерія"

Тернопіль, 2017

Конспект лекцій розроблений у відповідності за спеціальністю 123 „Комп’ютерна інженерія ”

Укладач :доцент кафедри комп’ютерних систем та мереж,
к.т.н. Тиш Є.В.

Рецензент:

Відповідальний за випуск: зав. каф. КС, к.т.н., доц.Осухівська Г.М.

Затверджено на засіданні кафедри комп’ютерних систем та мереж, протокол №____ від «___» _____2017 р.

Схвалено та рекомендовано до друку методичною комісією факультету комп’ютерно-інформаційних систем і програмної інженерії Тернопільського національного технічного університету імені Івана Пулюя, протокол №____ від «___» _____2017р.

Конспект лекцій складений з врахуванням методичних розробок інших вищих закладів освіти, а також матеріалів літературних джерел, перелічених в списку.

ЗМІСТ

ЛЕКЦІЯ 1

ВВЕДЕННЯ В ТЕОРІЮ СИСТЕМ ЧИСЛЕННЯ

Системою числення називається метод позначення (запису) чисел, який по-суті, є спеціальною мовою, алфавітом якої є множина символів, які називаються цифрами, а синтаксисом – правила, що дають змогу однозначно здійснити записування чисел. Записування числа в деякій системі числення називають кодом числа. Коротко число записується таким чином:

$$A = a_n a_{n-1} \ldots a_2 a_1 a_0 .$$

Окрему позицію в зображенні числа прийнято називати розрядом, а номер позиції – номером розряду. Число розрядів у записі числа називається *розрядністю*. У технічному аспекті розрядність інтерпретується як довжина розрядної сітки. Якщо алфавіт має p різних значень, то розряд a_i в числі розглядається як i-та цифра, якій може бути присвоєно одне із p значень.

Кожній цифрі a_i даного числа A однозначно відповідає її кількісний (числовий) еквівалент – $K(A)$. Кількісний еквівалент числа A, заданого у визначеній системі числення, є деякою функцією числових еквівалентів усіх його цифр, а саме:

$$K(A) = f[K(a_n), \ldots, K(a_1)].$$

Залежно від способу записування чисел та способу обчислення їх кількісного еквівалента системи числення поділяють на: непозиційні (алфавітні, ієрогліфічні) та позиційні (однорідні, неоднорідні).

Непозиційними системами числення називають такі системи числення, алфавіт яких має необмежену кількість символів (цифр), причому кількісний еквівалент будь-якої цифри сталий і залежить тільки від її графічного написання, але не від позиції в числі.

Ієрогліфічні системи числення – це такі непозиційні системи числення, в яких кожна цифра подана своїм символом, значком або ієрогліфом.

Алфавітні системи числення – це непозиційні системи числення, в яких буквам (всім або тільки деяким) приписуються числові значення, які, зазвичай, відповідають порядку букв в алфавіті. Переважно перші дев'ять букв отримують значення від 1 до 9, наступні дев'ять – від 10 до 90 й т.д. Для запису числа записуються букви, сума значень яких виражає це число.

Позиційними системами числення називають такі, алфавіт яких має скінчену кількість символів, причому значення кожної цифри в числі визначається не тільки її написанням, але й знаходиться в строгій залежності від позиції в записі числа. Позиційні системи числення мають ряд переваг відносно непозиційних, основною з яких є зручність виконання арифметичних операцій.

У загальному вигляді число A в позиційній системі числення може бути записано таким чином:

$$A = a_n p_{n-1} \ldots p_1 + a_{n-1} p_{n-2} \ldots p_1 + \ldots + a_2 p_1 + a_1, \qquad (1.1)$$

де a_i – цифра i-го розряду числа, причому $\{a_i = \overline{0, p_l - 1}\}$ є базою системи числення; p_l – основа системи числення; $p_i = \prod_{l=0}^{i} p_l$ – вага i-го розряду числа.

У *неоднорідних позиційних системах числення* p_l не залежать один від іншого та можуть набувати будь-які значення, ці системи ще називають системами зі змішаною основою. У таких системах числення в кожному i-му розряді кількість допустимих символів може бути різною, при цьому $0 \le a_i < p_{i-1}$, де p_{i-1} – основа системи числення в i-му розряді. Прикладом неоднорідної позиційної системи числення є система числення часу.

Однорідні позиційні системи числення є частинним випадком позиційних систем числення при $p_i = p_j$ для всіх i та j, тобто в них ваги окремих розрядів є рядом членів геометричної прогресії зі знаменником p. Тому число в однорідних системах може бути записано у вигляді поліному

$$A = a_n p^n + a_{n-1} p^{n-1} + \ldots + a_1 p^1 + a_0 p^0 + a_{-1} p^{-1} + \ldots + a_{-k} p^{-k} = \sum_{i=-k}^{n} a_i p^i , \quad (1.2)$$

де $a_i = \overline{0, p-1}$, а знаменник геометричної прогресії p має назву основи системи числення. Очевидно, що основою однорідної позиційної системи може бути будь-яке ціле число, оскільки в означенні позиційних систем числення не накладено ніяких обмежень на величину основи. Тому можлива нескінченна множина позиційних систем числення.

Переведення чисел з однієї системи числення в іншу.

Існує два основних методи переведення числа з однієї системи числення в іншу: табличний та розрахунковий.

Табличний метод прямого переведення заснований на використанні таблиць відповідності чисел різних систем числення. Цей метод надто громіздкий та вимагає значного об’єму пам’яті для зберігання таблиць, але може бути використаний для будь-яких систем числення (не тільки для позиційних). Суть іншого виду табличного методу полягає в тому, що існують таблиці еквівалентів у кожній системі тільки для цифр, тобто баз цих систем та степенів основ (додатних та від’ємних), тобто базису систем. Задача переведення зводиться до того, що у поліномі (1.2) для початкової системи числення подаються еквіваленти з нової системи для всіх цифр та їх ваг розрядів та проводять дії (множення та додавання) за правилами арифметики за новою основою *p*. Отриманий при цьому результат буде зображати число в новій системі числення. Цей метод використовується тільки до позиційних систем числення.

Розрахунковий метод використовується тільки до однорідних позиційних систем числення.

Правило переведення цілих чисел з однієї позиційної системи в іншу позиційну систему: щоб перевести ціле число з однієї позиційної системи числення в іншу, необхідно початкове число послідовно ділити на основу нової системи числення, що записане в початковій системі, до отримання частки, яка

дорівнює нулеві. Число в новій системі числення записується із залишків від ділення, починаючи з останнього.

Правило переведення правильний дробів з однієї позиційної системи в іншу позиційну систему: щоб перевести правильний дріб з однієї позиційної системи в іншу, необхідно початкове число послідовно множити на основу нової системи числення, записану в старій системі числення до отримання заданої точності. Дріб у новій системі числення запишеться у вигляді цілих частин добутків, починаючи з першої частини.

Правило переведення неправильний дробів з однієї позиційної системи в іншу позиційну систему: при переведенні неправильних дробів необхідно окремо перевести цілу та дробову частини числа за наведеними вище правилами переведення та записати в новій системі числення, залишивши незмінним положення коми.

ЛЕКЦІЯ 2

СИСТЕМИ ЧИСЛЕННЯ

2.1. Двійкова система числення.

Під *двійковою системою числення* розуміють таку систему, в якій для зображення чисел використовуються два символи, а ваги розрядів змінюються за законом $2^{\pm k}$ (де k – довільне ціле число). *Класичною двійковою системою* є система з символами 0, 1. Її двійкові цифри називають бітами.

Щоб опанувати будь-яку систему числення, потрібно вміти додавати та множити в ній будь-які цифри. Арифметичні операції в двійковій системі числення виконуються так само, як й в десятковій відповідно до таблиць порозрядних обчислень.

Таблиця 2.1 множень двійкових чисел повністю визначається двома правилами: 1) множення будь-якого числа на 0 дає 0; 2) множення будь-якого числа на 1 залишає його без зміни.

Для додавання є тільки одне правило, згідно з яким додавання 0 до будь-якого числа не змінює цього числа. Тоді таблиця додавання матиме вигляд згідно з таблицею 2.2.

Таблиця 2.1. Таблиця множення двійкових чисел

×	0	1
0	0	0
1	0	1

Таблиця 2.2. Таблиця додавання двійкових чисел

+	0	1
0	0	1
1	1	10

2.3. Форми подання чисел у цифрових автоматах.

Формою подання чисел у цифрових автоматах називають сукупність правил, що дають змогу встановити взаємну відповідність між записом числа та його кількісним еквівалентом.

Машинне (автоматне) зображення числа є поданням числа в розрядній сітці цифрового автомата. Умовне позначення машинного зображення числа, наприклад *A*, записується як [*A*]. Тоді справедливе співвідношення $A = [A]K_A$, де K_A – коефіцієнт, величина якого залежить від форми подання чисел в автоматі.

Через обмежену довжину машинних слів, множина чисел, яку можна подати в машині, скінченна. Порівняння різних форм подання чисел у комп'ютерах зазвичай робиться на основі порівняння *діапазону й точності подання числа*.

У повсякденній практиці найпоширенішою є форма подання чисел у вигляді послідовності цифр, що розділені комою на цілу та дробову частини. Числа, подані в такій формі, називають числами *з природною комою або числами в природній формі*. У природній формі число записується в природному натуральному вигляді, наприклад, 12560 – ціле число, 0,003572 – правильний дріб, 4,89760 – неправильний дріб.

При поданні чисел у такій формі обов'язково потрібно для кожного числа давати вказівку про положення його коми в розрядній сітці, виділеній для подання числа в машині, що вимагає додаткових апаратних витрат досить великого об'єму. Тому в комп'ютерах (цифрових автоматах) набули поширення дві інші форми подання чисел: *з фіксованою та плаваючою комою*.

Необхідність у вказані положення коми відпадає, якщо місце коми в розрядної сітки машини заздалегідь фіксовано раз та назавжди. Така форма подання чисел називають поданням *з фіксованою комою (крапкою)*.

Щоб спростити функціонування автомата, необхідно обмежити вхідну інформацію будь-якою однією областю чисел (на вхід автомата бажано подавати або цілі числа, або правильні дроби, або будь-які числа), це дасть змогу визначити величину масштабного коефіцієнта K_A. Наприклад, якщо на вхід цифрового автомата надходять тільки правильні дроби, то

$$-1 < [A]_\phi < 1, \qquad (2.3)$$

де $[A]_\phi$ – машинне зображення числа для форми представлення з фіксованою комою.

Тоді число A буде подане у вигляді $A = [A]_\phi K_A$.

Величина масштабного коефіцієнта K_A, що задовольняє умову (2.3), визначає той факт, що в машинному зображенні кома завжди стоїть після цілої частини дробу, тобто перед її старшим розрядом. Отже, можна зберігати тільки дробову частину числа (цифрову частину), а в розряді цілої частини писати додаткову інформацію.

Оскільки числа бувають додатні та від'ємні, то формат (розрядна сітка) машинного зображення розбивається на знакову частину та поле числа (рисунок 2.1). У полі числа розміщується саме зображення числа, що називається мантисою числа. Для кодування знака числа використовується найстарший розряд розрядної сітки, відведеної для зображення двійкового числа, а інші розряди відводяться під мантису числа.

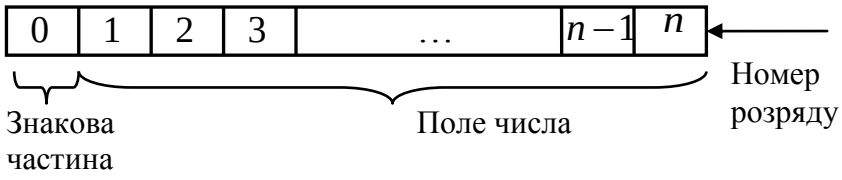


Рисунок 2.1. Подання чисел у форматі з фіксованою комою

Положення коми в розрядній сітці строго фіксується, зазвичай або правіше самого молодшого розряду мантиси, або лівіше від найстаршого. У першому випадку число подається як ціле, в другому – як правильний дріб. Нині в переважній більшості в комп'ютерах у форматі з фіксованою комою подаються цілі числа.

У знакову частину записується інформація про знак числа. Прийнято, що знак позитивного числа "+" зображується символом 0, а знак від'ємного числа "–" – символом 1.

Для машинного подання від'ємних чисел використовують *прямі, додаткові та зворотні коди*. Спрощене означення цих кодів може бути дане таким чином. Якщо число A в звичайному двійковому коді – *прямому двійковому коді*, зображувати як $[A]_{пр} = 0, a_1 a_2 \ldots a_n$, тоді число „$-A$" в цьому самому коді подається як $[A]_{пр} = 1, a_1 a_2 \ldots a_n$, а в *зворотному* (*інверсному*) *коді* це число буде мати вигляд $[A]_{зв} = 1, \bar{a}_1 \bar{a}_2 \ldots \bar{a}_n$, де $\bar{a}_i = 1$, якщо $a_i = 0$ та $\bar{a}_i = 0$, якщо $a_i = 1$ (a_i – цифра i-того розряду двійкового числа). Отже, при переході від прямого коду до зворотного всі цифри розрядів мантиси числа інвертуються.

Додатковий код числа $A = 0, a_1 a_2 \ldots a_n$ – це таке машинне зображення цього числа $[A]_{д} = 1, \bar{a}_1 \bar{a}_2 \ldots \bar{a}_n$, для якого $\bar{a}_i = 0$, якщо $a_i = 1$ та $\bar{a}_i = 1$, якщо $a_i = 0$, за винятком останнього значущого розряду, для якого $\bar{a}_k = 1$ при $\bar{a}_k = 1$. Наприклад, число $A = -0{,}101110$ запишеться в додатковому коді як $[A]_{д} = 1{,}010010$.

Таким чином, для отримання додаткового коду від'ємних чисел необхідно спочатку інвертувати цифрову частину початкового числа, внаслідок чого виходить його зворотний код, а потім додати одиницю в молодший розряд цифрової частини числа.

Наголосимо, що *додатковий та зворотний коди використовуються тільки для подання від'ємних двійкових чисел у формі з фіксованою комою*. Додатні числа в цих кодах не змінюють свого зображення та подаються як у прямому коді.

2.4. Форма подання чисел з плаваючою комою

У нормальній формі

$$A_н = m_A p_A, \tag{2.4}$$

де m_A – мантиса числа A; p_A – порядок числа A (характеристика числа).

Таке подання чисел не є однозначним, тому для визначеності зазвичай вводять деякі обмеження. Найпоширеніше та зручне для подання в цифрових автоматах обмеження виду

$$q^{-1} \le |m_A| < 1, \tag{2.5}$$

де q – основа системи числення.

Нормальна форма подання чисел – форма подання чисел, для якої справедлива умова (2.5).

Оскільки в цьому випадку абсолютне значення мантиси лежить у межах від q^{-1} до $1 - q^{-n}$, де n – кількість розрядів для зображення мантиси без знака, положення розрядів числа в його автоматному зображенні не постійне. Тому таку форму подання чисел називають також *формою подання з плаваючою комою*. Формат машинного зображення числа з плаваючою комою повинен мати знакові частини та поля для мантиси й порядку (рисунок 2.2). Виділяються спеціальні розряди для зображення знака числа (мантиси) та знака порядку або характеристики (рисунок 2.2). Кодування знаків залишається таким самим, як було з фіксованою комою.

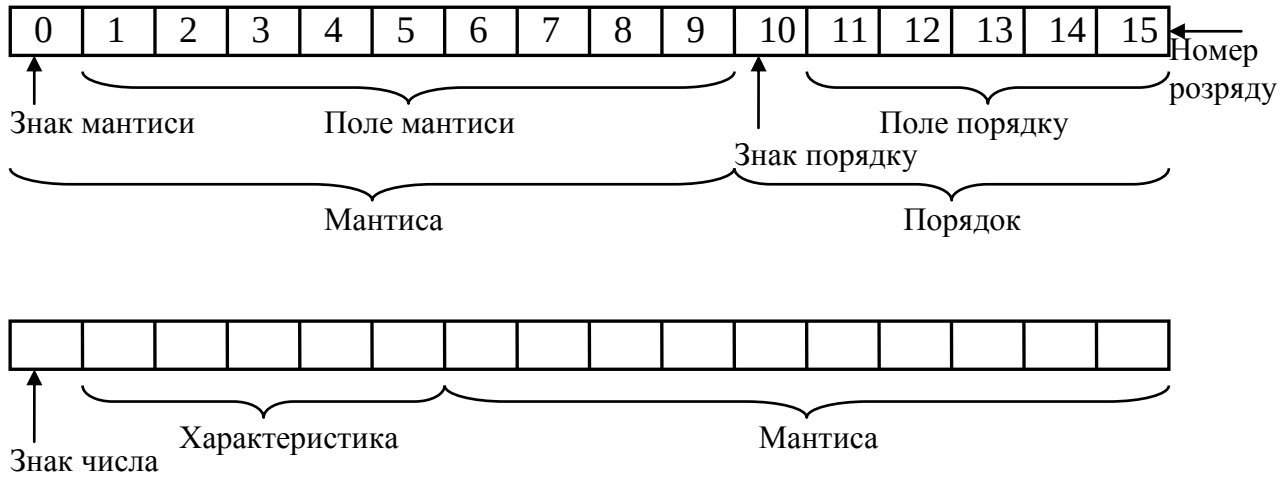


Рисунок 2.2. Подання чисел у формі з плаваючою комою

ЛЕКЦІЯ 3

АРИФМЕТИЧНІ ТА ЛОГІЧНІ ОПЕРАЦІЇ НАД ЧИСЛАМИ У ДВІЙКОВІЙ СИСТЕМІ ЧИСЛЕННЯ. ДОДАВАННЯ ДВІЙКОВИХ ЧИСЕЛ.

3.1. Формальні правила двійкової арифметики

Важливою функцією більшості обчислювальних пристроїв є виконання арифметичних операцій. У зв'язку з цим у комп'ютерах (цифрових автоматах) виділяють спеціальний функціональний блок – арифметико-логічний пристрій, призначений для виконання операцій над числовими кодами, та який у комп'ютерах переважно є складовою процесора.

При виконанні арифметичних дій завжди беруть участь два та більше числа. У результаті арифметичної операції з'являється нове число

$$C = A \nabla B, \tag{3.1}$$

де ∇ – знак арифметичної дії (додавання, віднімання, множення, ділення).

Числа, які беруть участь у арифметичних операціях, що виконуються цифровими автоматами, називають *операндами*.

Оскільки цифровий автомат оперує тільки автоматними зображеннями чисел, то для машинних операцій вираз (3.1) правильніше записати у вигляді

$$[C] = [A] \nabla [B], \tag{3.2}$$

де в квадратних дужках [ ] – позначення автоматних зображень операндів.

Розглянемо формальні правила виконання арифметичних операцій додавання та віднімання на рівні розрядів операторів.

Правила порозрядних дій при додаванні операндів A та B можна записати так:

$$a_i + b_i + П_{i-1} = c_i + П_i, \tag{3.3}$$

де $П_{i-1}$ – перенесення з (i-1)-го розряду; $П_i$ – перенесення в (i+1)-ший розряд (переноси набувають значення 0 або 1).

Двійковий напівсуматор – пристрій, що виконує арифметичні дії за правилами, вказаними в таблиці3.1.

Таблиця 3.1. Правила додавання двійкових цифр

a_i	b_i	c_i	$П_i$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Таблиця 3.2. Правила додавання двійкових цифр з урахуванням переносу

a_i	b_i	$П_{i-1}$	c_i	$П_i$
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Двійковий суматор – пристрій, що виконує арифметичні дії за правилами, вказаними в таблиці 3.2.

На основі правил двійкової арифметики можна записати правила віднімання двійкових чисел так, як це показано в таблиці 3.3, де z_{i+1} – запозичення зі старшого розряду. Запозичення рівносильне відніманню одиниці зі старшого розряду. З урахуванням одиниці запозичення зі старшого сусіднього розряду, правила віднімання двійкових чисел можна записати так, як це показано в таблиці 3.4 (щоб відрізняти запозичення від перенесення, перед одиницею поставлений знак мінус).

Таблиця 3.3. Правила віднімання двійкових цифр

a_i	b_i	c_i	z_{i+1}
0	0	0	0
1	0	1	0
1	1	0	0
0	1	1	-1

Таблиця 3.4. Правила додавання двійкових цифр з урахуванням запозичення

a_i	b_i	z_i	c_i	z_{i+1}
0	0	0	0	0
1	0	0	1	0
1	1	0	0	0
0	1	0	1	-1
0	0	-1	1	-1
1	0	-1	0	0
1	1	-1	1	-1
0	1	-1	0	-1

Якщо A – зменшувальне (перший операнд), B – віднімальне (другий операнд), то для порозрядних дій справедлива рівність

$$a_i + b_i + z_i = c_i + z_{i+1}. \qquad (3.4)$$

3.2. Додавання двійкових чисел

Арифметичні операції можна виконувати з двійковими числами, поданими в прямому, зворотному та додатковому кодах. Якщо операнди подані в прямому коді та мають однакові знаки, то над ними при алгебраїчному додаванні природно виконується процедура додавання. Якщо ж операнди мають різні знаки – процедура віднімання. Як вже зазначалося, для спрощення апаратних засобів комп'ютера процедура віднімання замінюється додаванням завдяки тому, що від'ємний операнд подається у зворотному або додатковому коді.

Алгебраїчне додавання чисел, поданих у формі з фіксованою комою. Розглянемо кілька видів двійкових суматорів.

Двійковий суматор прямого коду (ДСПК) – суматор, у якому відсутній ланцюг порозрядного переносу між старшими цифровими та знаковими розрядами. На ДСПК можна додавати тільки числа, що мають однакові знаки, тобто такий суматор не може виконувати операцію алгебраїчного додавання.

Нехай задані операнди

$$[A]_{пр} = Sg_A a_1 a_2 \ldots a_n,\ [B]_{пр} = Sg_B b_1 b_2 \ldots b_n,$$

де Sg_A, Sg_B – відповідно вміст знакових розрядів автоматних зображень для A та B (символ походить від англійського слова *sign* – знак); a_i, b_i – цифрові розряди зображень.

Якщо $Sg_A = Sg_B$, то сума чисел буде мати знак будь-якого з доданків, а цифрова частина результату буде отримана після додавання цифрових частин операндів.

Приклад 3.1. Додати числа $A = 0{,}0101$ та $B = 0{,}1010$ на ДСПК.

$$\begin{array}{ll} [A]_{пр} = 0{,}0101 & Sg_A = 0, \\ \overset{+}{[B]_{пр}} = 0{,}1010 & Sg_B = 0, \\ \hline [C]_{пр} = 0{,}1111 & \overline{Sg_C = 0}. \end{array}$$

Приклад 3.2. Додати числа $A = -0{,}0100$ та $B = -0{,}0110$ на ДСПК.

$$\begin{array}{ll} [A]_{пр} = 1{,}0100 & Sg_A = 1, \\ \overset{+}{[B]_{пр}} = 1{,}0110 & Sg_B = 1, \\ \hline [C]_{пр} = 1{,}1010 & \overline{Sg_C = 1}. \end{array}$$

При додаванні чисел на ДСПК можливий випадок, коли абсолютне значення суми операндів перевищує одиницю. Тоді кажуть, що має місце переповнення розрядної сітки автомата. Ознака переповнення – наявність одиниці перенесення зі старшого розряду цифрової частини суматора. У цьому випадку генерується сигнал переповнення $\varphi = 1$, за яким відбувається автоматична зупинка машини та коректування масштабних коефіцієнтів з таким розрахунком, щоб запобігти появі переповнення.

Двійковий суматор додаткового коду (ДСДК) – суматор, що оперує зображеннями чисел у додатковому коді. Характерна риса ДСДК – наявність ланцюга порозрядного перенесення зі старшого розряду цифрової частини в знаковий розряд. Правило додавання чисел на ДСДК формулюють так: *сума додаткових кодів чисел є додатковий код результату*. Правило справедливе для всіх випадків, у яких не виникає переповнення розрядної сітки, що дає змогу складати автоматні зображення чисел за правилами двійкової арифметики (таблиця 3.2), не розділюючи знакову та цифрову частини автоматного зображення числа.

Приклад 3.3. Додати числа $A = 0{,}0010$ та $B = 0{,}1100$ на ДСДК.

$$\begin{array}{l} [A]_{д} = 0{,}0010 \\ \overset{+}{[B]_{д}} = 0{,}1100 \\ \hline [C]_{д} = 0{,}1110. \end{array}$$

Приклад 3.4. Додати числа $A = -0{,}0110$ та $B = 0{,}0100$ на ДСДК.

$$\begin{array}{l} [A]_{д} = 1{,}1010 \\ \overset{+}{[B]_{д}} = 0{,}0100 \\ \hline [C]_{д} = 1{,}1111. \end{array}$$

Приклад 3.5. Додати числа $A = 0{,}1011$ та $B = -0{,}0110$ на ДСДК.

$$[A]_{\partial} = 0{,}1011$$
$$+$$
$$[B]_{\partial} = 1{,}1010$$
$$[C]_{\partial} = 0{,}0101.$$

Двійковий суматор зворотного коду (ДСЗК) – суматор, що оперує зображеннями чисел у зворотному коді. Характерна особливість ДСДК – наявність ланцюга циклічного перенесення зі знакового розряду в молодший розряд цифрової частини. Правило додавання чисел на ДСЗК формулюють так: *сума зворотних кодів чисел є зворотний код результату*. На ДСЗК машинні зображення чисел додаються за правилами, поданими в таблиці 3.2.

Приклад 3.6. Додати числа $A = 0{,}1010$ та $B = 0{,}0011$ на ДСЗК.

$$[A]_{зв} = 0{,}1010$$
$$+$$
$$[B]_{зв} = 0{,}0011$$
$$[C]_{зв} = 0{,}1101.$$

Приклад 3.7. Додати числа $A = -0{,}0011$ та $B = 0{,}0101$ на ДСЗК.

$$[A]_{зв} = 1{,}1100$$
$$+$$
$$[B]_{зв} = 0{,}0101$$
$$0{,}0001$$
$$+$$
$$1$$
$$[C]_{зв} = 0{,}0010.$$

Приклад 3.8. Додати числа $A = 0{,}0101$ та $B = -0{,}0111$ на ДСЗК.

$$[A]_{зв} = 0{,}0101$$
$$+$$
$$[B]_{зв} = 1{,}1000$$
$$[C]_{зв} = 1{,}1101.$$

Приклад 3.9. Додати числа $A = -0{,}0101$ та $B = -0{,}1000$ на ДСЗК.

$$[A]_{зв} = 1{,}1010$$
$$+$$
$$[B]_{зв} = 1{,}0111$$
$$1{,}0001$$
$$+$$
$$1$$
$$[C]_{зв} = 1{,}0010.$$

Переповнення розрядної сітки. При додаванні чисел однакового знака, поданих у формі з фіксованою комою, у цифрових автоматах може виникнути переповнення розрядної сітки.

1. Ознакою переповнення розрядної сітки при додаванні чисел у прямому коді є поява одиниці перенесення зі старшого розряду цифрової частини числа.

2. Ознака переповнення розрядної сітки при додаванні чисел у додатковому та зворотному кодах – отримання знака результату, протилежного до знаків операндів.

Для виявлення переповнення розрядної сітки у складі цифрового автомата мають бути передбачені апаратні засоби, які автоматично генерують ознаку переповнення – сигнал переповнення φ.

Один із методів виявлення переповнення розрядної сітки передбачає введення допоміжного розряду в знакову частину зображення числа (рисунок 3.3), який називають *розрядом переповнення*. На рисунках 3.4 та 3.5 відповідно подано зображення додатного та від'ємного чисел. Таке зображення числа називають *модифікованим*. Тоді у випадку появи переповнення сигнал $\varphi = 1$,

$$Sg_1 \cap \bar{S}g_2 = 1,\ \bar{S}g_1 \cap Sg_2 = 1, \qquad (3.5)$$

в інших випадках $\varphi = 0$.

Алгебраїчне додавання чисел, які подано у формі з плаваючою комою. При виконанні будь-яких арифметичних дій над операндами, поданими у формі з плаваючою комою, операції, що виконуються над мантисами та порядками (чи характеристиками) цих чисел, різні. Тому перед початком будь-якої арифметичної процедури кожен з операндів "розрізається": порядок (характеристика) відділяється від мантиси операнда, щоб можна було над ними виконувати необхідні окремі процедури. Після виконання конкретної арифметичної дії та обов'язкової процедури нормалізації результату, його порядок або характеристика та мантиса "склеюються" у звичайний формат з плаваючою комою.

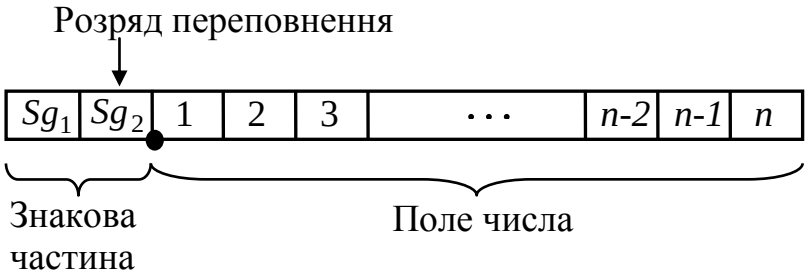


Рисунок 3.3. Зображення числа в модифікованому коді

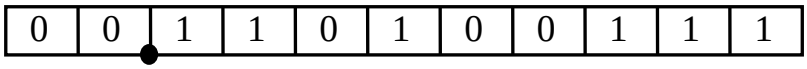


Рисунок 3.4. Зображення додатного числа в модифікованому коді

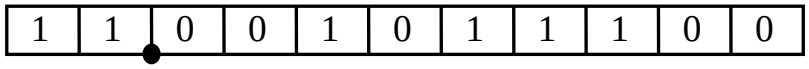


Рисунок 3.5. Зображення від'ємного числа в модифікованому коді

Оскільки для чисел з плаваючою комою справедлива умова (2.5), то будь-який результат, що не задовольняє цю умову, має бути приведений відповідно до формули (2.5). Таку операцію називають *нормалізацією числа*. Операція нормалізації числа полягає в перевірці умови (2.5) та зсуву автоматного зображення мантиси в той чи інший бік. Зсуви можуть здійснюватися на один розряд та більше, в лівий або правий бік у межах розрядної сітки машини.

Простий зсув – операція, що виконується за правилами таблиці 3.5.

Модифікований зсув – операція над модифікованими зображеннями, яка виконується згідно з таблицею 3.6 (величина α залежить від коду: для додаткового коду $\alpha = 0$, для зворотного $\alpha = 1$).

Порушення нормалізації числа – невиконання умови (2.5). Оскільки умова (2.5) містить дві нерівності, то може бути порушення справа та зліва. Ознакою порушення нормалізації числа справа γ (коли величина результату дорівнює або перевищує одиницю) є наявність різнойменних комбінацій у знакових розрядах суматора, тобто

$$\gamma = 1, \text{ якщо } \bar{S}g_1 \wedge Sg_2 = 1; Sg_1 \wedge \bar{S}g_2 = 1, \tag{3.6}$$

(в інших випадках $\gamma = 0$), де γ – ознака порушення нормалізації числа справа, що вказує на необхідність зсуву числа вправо на один розряд.

Таблиця 3.5. Операція простого зсуву

Вхідна комбінація	Зсунута вліво на один розряд	Зсунута вправо на один розряд
$0, a_1a_2 \ldots a_n$	$a_1, a_2 \ldots a_n 0$	$0, 0a_1 \ldots a_{n-1}$
$1, a_1a_2 \ldots a_n$	$a_1, a_2 \ldots a_n \alpha$	$0, 1a_1 \ldots a_{n-1}$

Таблиця 3.6. Операція модифікованого зсуву

Вхідна комбінація	Зсунута вліво на один розряд	Зсунута вправо на один розряд
$00, a_1a_2 \ldots a_n$	$0a_1, a_2 \ldots a_n 0$	$00, 0a_1 \ldots a_{n-1}$
$01, a_1a_2 \ldots a_n$	$1a_1, a_2 \ldots a_n 0$	$00, 1a_1 \ldots a_{n-1}$
$10, a_1a_2 \ldots a_n$	$0a_1, a_2 \ldots a_n \alpha$	$1, 0a_1 \ldots a_{n-1}$
$11, a_1a_2 \ldots a_n$	$1a_1, a_2 \ldots a_n \alpha$	$1, 1a_1 \ldots a_{n-1}$

Ознакою порушення нормалізації числа зліва δ (коли результат за власною величиною виявляється менше $1/q$) є наявність однакових комбінацій у розряді переповнення та старшому розряді цифрової частини суматора (p_1):

$$\delta = 1, \text{ якщо } Sg_2 \wedge p_1 = 1; \bar{S}g_2 \wedge \bar{p}_1 = 1 \tag{3.7}$$

(в інших випадках $\delta = 0$), де δ – ознака порушення нормалізації, що вказує на необхідність зсуву числа вліво на один розряд.

Таким чином, операція нормалізації числа складається з сукупності зсувів та перевірки наявності ознак порушення γ та δ.

Розглянемо додавання чисел $A = m_A p_A$ та $B = m_B p_B$, що мають однаковий порядок $p_A = p_B$. Обидві мантиси задовольняють умову нормалізації.

Додавання мантис здійснюється на відповідному суматорі за правилами, викладеним раніше для чисел, які подано у формі з фіксованою комою. Якщо після додавання мантиса результату задовольняє умову нормалізації (тобто $\delta = 0, \gamma = 0$), то до цього результату приписується порядок будь-якого з операндів. В іншому випадку – проходить нормалізація числа.

ЛЕКЦІЯ 4

АРИФМЕТИЧНІ ТА ЛОГІЧНІ ОПЕРАЦІЇ НАД ЧИСЛАМИ У ДВІЙКОВІЙ СИСТЕМІ ЧИСЛЕННЯ. МНОЖЕННЯ ДВІЙКОВИХ ЧИСЕЛ.

4.1. Множення двійкових чисел.

У двійковій системі числення операція множення може виконуватися у два способи.

1. Множення, починаючи з молодших розрядів множника,

```
  ×1001
   1010
 ------
   0000
  1001
 0000
1001
------
1011010.
```

2. Множення, починаючи зі старших розрядів множника,

```
× 1001
  1010
------
 1001
  0000
   1001
    0000
--------
 1011010.
```

В обох випадках операція множення складається з ряду послідовних операцій додавання частинних добутків. Операціями додавання управляють розряди множника: якщо в якомусь розряді множника знаходиться одиниця, то до суми частинних добутків додається множене з відповідним зсувом (вліво або вправо), якщо в розряді множника – нуль, то множене не додається, але враховується, що у подальшій операції аналізу розряду множника потрібно зробити додатковий зсув.

Якщо, наприклад, у наступному після нульового розряду множника зустрічається 1, то множене зсувається на 2 розряди та додається до суми частинних добутків. Скільки підряд зустрічатиметься 0, – стільки додаткових зсувів множеного потрібно буде зробити, коли в черговому розряді зустрінеться 1, а потім додавати множене до суми частинних добутків.

Таким чином, окрім операції додавання чисел для отримання добутку потрібна операція зсуву чисел. При цьому з’являється можливість зсувати множене на суму частинних добутків, що дає підставу для різних методів реалізації операції множення.

Метод 1. Нехай A – множене ($A>0$), B – множник ($B>0$), C – добуток. Тоді у випадку подання чисел у формі з плаваючою комою отримаємо: $A=0,a_1a_2\ldots a_n;\ B=0,b_1b_2\ldots b_n=b_1\cdot 2^{-1}+b_2\cdot 2^{-2}+\ldots+b_n\cdot 2^{-n}$.

Звідси

$$\begin{aligned}C=AB&=0,a_1a_2\ldots a_n\left(b_1\cdot 2^{-1}+b_2\cdot 2^{-2}+\ldots+b_n\cdot 2^{-n}\right)=\\&=\left(2^{-1}\cdot 0,a_1a_2\ldots a_n\right)\cdot b_1+\left(2^{-2}\cdot 0,a_1a_2\ldots a_n\right)\cdot b_2+\ldots+\left(2^{-n}\cdot 0,a_1a_2\ldots a_n\right)\cdot b_n.\end{aligned}\tag{4.1}$$

Множення на 2^{-n} означає зсув на n розрядів вправо числа, яке знаходиться в дужках, тобто в даному випадку зсувається вправо множене та множення починається зі старших розрядів.

Метод 2. Нехай $A=0,a_1a_2\ldots a_n$ – множене, $B=0,b_1b_2\ldots b_n$ – множник.

Множник можна перетворити, використовуючи метод Горнера,

$$B=\left(\ldots\left(\left(b_n\cdot 2^{-1}+b_{n-1}\right)\cdot 2^{-1}+\ldots+b_2\right)\cdot 2^{-1}+b_1\right)\cdot 2^{-1}.$$

Тоді

$$\begin{aligned}C=AB=\left(\ldots\left(\left(b_n\cdot 0,a_1a_2\ldots a_n\right)\cdot 2^{-1}+b_{n-1}\cdot 0,a_1a_2\ldots a_n\right)\cdot 2^{-1}+\\+\ldots+b_1\cdot 0,a_1a_2\ldots a_n\right)\cdot 2^{-1}.\end{aligned}\tag{4.2}$$

Тут множення починається з молодших розрядів та зсувається вправо сума частинних добутків.

Метод 3. Нехай $A=0,a_1a_2\ldots a_n$ – множене, $B=0,b_1b_2\ldots b_n$ – множник.

Множник, використовуючи метод Горнера, можна записати так:

$$\begin{aligned}B&=2^{-n}\left(b_1\cdot 2^{n-1}+b_2\cdot 2^{n-2}+\ldots+b_{n-1}\cdot 2^1+b_n\cdot 2^0\right)=\\&=2^{-n}\left(\ldots\left(b_1\cdot 2^1+b_2\right)2^1+\ldots+b_{n-1}\cdot 2^1+b_n\right).\end{aligned}$$

У цьому випадку

$$\begin{aligned}C=AB=2^{-n}\left(b_n\cdot 0,a_1a_2\ldots a_n+\left(2^1\cdot 0,a_1a_2\ldots a_n\right)b_{n-1}+\right.\\\left.+\ldots+\left(2^{n-1}\cdot 0,a_1a_2\ldots a_n\right)\cdot b_1\right).\end{aligned}\tag{4.3}$$

Це означає, що множення починається з молодших розрядів, множене зсувається вліво на один розряд в кожному такті.

Метод 4. Нехай $A=0,a_1a_2\ldots a_n$ – множене, $B=0,b_1b_2\ldots b_n$ – множник.

Якщо множник B записати за методом Горнера

$$\begin{aligned}C=AB=2^{-n}\left(\left(\ldots\left(2^1\left(b_1\cdot 0,a_1a_2\ldots a_n\right)+b_2\cdot 0,a_1a_2\ldots a_n\right)2^1+\right.\\\left.+\ldots+b_{n-1}\cdot 0,a_1a_2\ldots a_n\right)2^1+b_{n-1}\cdot 0,a_1a_2\ldots a_n\right),\end{aligned}\tag{4.4}$$

то множення починається зі старшого розряду та в кожному такті зсувається вліво сума частинних добутків.

Аналіз формул (4.1)-(4.4) показує, що з формальної точки зору процес множення двох чисел може бути поданий:

- при послідовному виконанні – у вигляді багаторазового повторення за кількістю розрядів циклу

$$S_i=S_{i-1}+Ab_i,\tag{4.5}$$

де S_{i-1}, S_i – суми частинних добутків на $(i-1)$-му та i-му кроках відповідно;

- при паралельному виконанні – сумою членів діагональної матриці, для якої задані в рядках $A \cdot 2^i$, в стовпцях – b_i, де i – поточний номер розряду.

Множення чисел, які подано у формі з фіксованою комою, на двійковому суматорі прямого коду. Нехай задані машинні зображення двох чисел $[A]_{np} = Sg_A, a_1 a_2 \ldots a_n$ та $[B]_{np} = Sg_B, b_1 b_2 \ldots b_n$. Тоді їх добуток $[C]_{np} = Sg_C, c_1 c_2 \ldots c_n$, де $Sg_C = Sg_A \oplus Sg_B$ ($\oplus$ – знак операції додавання за модулем 2, що дорівнює $0 \oplus 0 = 0$, $1 \oplus 0 = 1$, $0 \oplus 1 = 1$, $1 \oplus 1 = 0$).

Приклад 4.1. Перемножити числа $[A]_{np} = 1,11010$ та $[B]_{np} = 0,11001$.

Знак добутку визначаємо окремо від цифрової частини відповідно до рівняння $Sg_C = Sg_A \oplus Sg_B = 1 \oplus 0 = 1$. При множенні будемо використовувати метод 1:

$$
\begin{array}{r}
\times\ 11010 \\
\quad 11001 \\
\hline
11010 \\
\quad 11010 \\
\quad\ 00000 \\
\quad\quad 00000 \\
\quad\quad\ 11010 \\
\hline
1010001010.
\end{array}
$$

Отже, $[C]_{np} = 1,1010001010$.

Множення чисел, які подано у формі з фіксованою комою, на двійковому суматорі додаткового коду. У випадках, коли числа в машині зберігаються в додаткових кодах, доцільно всі операції над числами проводити на суматорах додаткового коду. Але при цьому виникає низка особливостей, які необхідно враховувати.

Добуток додаткових кодів множників дорівнює додатковому коду результату тільки у випадку додатного множника.

Нехай множене A – будь-яке число, тобто $A = [A]_\partial$, а множник $B > 0$. Тоді

$$AB = [A]_\partial \cdot 0, b_1 b_2 \ldots b_n = [A]_\partial b_1 \cdot 2^{-1} + [A]_\partial b_2 \cdot 2^{-2} + \ldots + [A]_\partial b_n \cdot 2^{-n}. \quad (4.6)$$

На основі правила про додавання додаткових кодів можна стверджувати, що в правій частині рівняння (4.6) стоїть додатковий код результату.

Таким чином, множення на суматорі додаткового коду полягає в аналізі розрядів множника при $b_i = 1$ та в додаванні додаткового коду множеного до вмісту суматора. При цьому мають здійснюватися модифіковані зсуви.

Тепер розглянемо випадок, коли A – будь-яке число, а множник $B < 0$. Тоді $[B]_\partial = 1, \bar{b}_1 \bar{b}_2 \ldots \bar{b}_n$.

На основі того, що додатковий код є математичним доповненням основі системи числення ($|B|+[B]_д=q$, де $|B|$ – абсолютне значення числа B), то можна записати, що $B=[B]_д-2$, або $B=0,\bar{b}_1\bar{b}_2\ldots\bar{b}_n$. Отже, добуток чисел

$$AB=A(0,b_1b_2\ldots b_{n-1})=A\cdot 0,b_1b_2\ldots b_n-A. \qquad (4.7)$$

Формула (4.7) показує, що при від'ємному множнику добуток додаткових кодів операндів не дорівнює додатковому коду результату. Якщо ввести заміну $-A$ на $\bar{A}$, то можна вивести наступне правило. *Якщо множник від'ємний, то добуток чисел на суматорі додаткового коду отримується додаванням поправки* $[\bar{A}]$ *до добутку додаткових кодів множників*.

Множення чисел на двійковому суматорі зворотного коду. Розглянемо правила множення операндів, заданих у зворотному коді.

Добуток зворотних кодів множників дорівнює зворотному коду результату тільки у випадку додатного множника.

Нехай множене $A=[A]_{зв}$, а множник $B>0$. Тоді

$$AB=[A]_{зв}\cdot 0,b_1b_2\ldots b_n=[A]_{зв}b_1\cdot 2^{-1}+[A]_{зв}b_2\cdot 2^{-2}+\ldots+[A]_{зв}b_n\cdot 2^{-n}. \qquad (4.8)$$

За правилом про додавання зворотних кодів у правій частині даного рівняння отримується зворотний код результату.

Отже, множення на суматорі зворотного коду також полягає в аналізі розрядів множника, та якщо виявиться, що черговий розряд множника дорівнює одиниці, то до вмісту суматора додається зворотний код множеного.

Нехай $A=[A]_{зв}$ та $B<0$. Тоді $[B]_{зв}=1,b_1b_2\ldots b_n$. Відповідно до правил перетворення чисел у зворотний код $[B]_{зв}=2+B-2^{-n}$. Отже, $B=0,b_1b_2\ldots b_n+2^{-n}-1$.

$$AB=[A]_{зв}\cdot 0,b_1b_2\ldots b_n+[A]_{зв}\cdot 2^{-n}+\bar{A}. \qquad (4.9)$$

На основі (4.9) можна сформулювати наступне правило. *Якщо множник від'ємний, то добуток чисел на суматорі зворотного коду отримується додаванням поправок* $[\bar{A}]$ *та* $[A]_{зв}\cdot 2^{-n}$ *до добутку зворотних кодів множників*.

Множення чисел, які подано у формі з плаваючою комою. Для чисел, які подано у формі з плаваючою комою, обов'язковим є їх подання у вигляді мантиси та порядку (характеристики). При операції множення дії, що виконуються над мантисами та порядками, різні: мантиси перемножуються, порядки додаються. Очевидно, що результат множення може виявитися ненормалізованим, тоді буде необхідна нормалізація з відповідною корекцією порядку результату.

Приклад 4.2. Перемножити числа $A=-0{,}11001\cdot 2^{-3}$ та $B=0{,}10011\cdot 2^{+1}$, що задані в прямому коді.

Мантиси перемножуємо за правилами, що розглянуті для чисел, поданих у формі з фіксованою комою. Для перемноження мантис використаємо суматор прямого коду, а для додавання порядків – суматор зворотного коду.

Спочатку записуємо машинне зображення чисел

$$[m_A]_{пр} = 1{,}11001; [p_A]_{зв} = 1{,}100;$$
$$[m_B]_{пр} = 0{,}10011; [p_B]_{зв} = 0{,}001.$$

Мантиса добутку дорівнює $[m_C]_{пр} = 1{,}0111011011$. Додамо порядки

$$[p_C]_{зв} = [p_A]_{зв} + [p_B]_{зв} = 1{,}100 + 0{,}001 = 1{,}101.$$

Оскільки мантиса результату не задовольняє умову нормалізації (порушена ліва границя: $\delta = 1,\ \gamma = 0$), то проводиться зсув мантиси вліво на один розряд $[m'_C]_{пр} = 1{,}1110110110$, та корекція порядку

$$[p'_C]_{зв} = [p_C]_{зв} + 1{,}110 = 1{,}101 + 1{,}110 = 1{,}110.$$

Якщо суматор мантис містить тільки n розрядів, то після заокруглення отримаємо результат $C = -0{,}11110 \cdot 2^{-3}$.

ЛЕКЦІЯ 5

АРИФМЕТИЧНІ ТА ЛОГІЧНІ ОПЕРАЦІЇ НАД ЧИСЛАМИ У ДВІЙКОВІЙ СИСТЕМІ ЧИСЛЕННЯ. ДІЛЕННЯ ДВІЙКОВИХ ЧИСЕЛ.

Ділення двійкових чисел, які подано у формі з фіксованою комою. Ділення двійкових чисел багато в чому аналогічне діленню десяткових чисел. Алгоритм ділення полягає в тому, що дільник на кожному кроці віднімається від діленого стільки разів (починаючи зі старших розрядів), скільки це можливо для отримання найменшого додатного залишку. Тоді в черговий розряд частки записується цифра, що дорівнює числу дільників, які містяться в діленому на цьому кроці. Інакше кажучи, при діленні операцію віднімання повторюють до тих пір, поки зменшуване не стане менше за те, що віднімається. Число цих повторень показує, скільки разів від'ємник укладається в зменшуваному.

Приклад 5.1. Поділимо $A = 11001100_{\ 2}$ на $B = 1100_{\ 2}$.

ділене 11001100 | 1100
дільник 1100 | 10001
00001
- 0
11
- 0
110
- 0
1100
- 1100
0000.

Тут цифри частки отримуються послідовно, починаючи зі старшого розряду, шляхом віднімання дільника з отриманого залишку. Якщо отриманий додатний залишок, то цифра частки дорівнює одиниці; якщо залишок від'ємний, то цифра частки дорівнює нулеві, при цьому відновлюється попередній додатний залишок.

У разі додатного залишку для отримання наступної цифри останній залишок зсувається вліво на один розряд (або дільник управо на один розряд) та з нього віднімається дільник й т. д.

У разі від'ємного залишку відновлюється попередній додатний залишок додаванням до від'ємного залишку дільника та відновлений залишок зсувається на один розряд вліво (або зсувається дільник вправо на один розряд) та з нього віднімається дільник. Такий алгоритм отримав дістав назву алгоритму ділення з відновленням залишку. Формально всі дії можна описати таким чином.

Нехай A – ділене, B – дільник та C – частка, при цьому $A = 0, a_1 a_2 \ldots a_n$, $B = 0, b_1 b_2 \ldots b_n$, $C = 0, c_1 c_2 \ldots c_n$.

Для реалізації алгоритму ділення двійкових чисел, поданих у формі з фіксованою комою, необхідно, щоб виконувалась умова $|A| < |B|$. Якщо ця умова

не виконується, то на першому кроці виникає переповнення розрядної сітки та операція не виконується. Якщо $|A| < |B|$, то на першому кроці операції проводиться зсув дільника та визначається залишок $A_i = A - B \cdot 2^{-i}$.

Нехай $A_1 > 0$, тоді $C_1 = 1$. Процес ділення триває далі: $A_2 = A_1 - B \cdot 2^{-2}$.

Нехай $A < 0$, тоді $C_1 = 0$ та робиться відновлення залишку A_1: $A_2 = A_2 + B \cdot 2^{-2}$.

Цей залишок приймається за A_2' та ділення триває далі таким чином: $A_3 = A_2' - B \cdot 2^{-3}$.

Отже, алгоритм ділення можна описати в загальному вигляді на i-му кроці

$$A_i = A_{i-1} - B \cdot 2^{-i} \ , \tag{4.10}$$

якщо $A_i \geq 0$, то $C_i = 1$ та перехід до наступного кроку;
якщо $A_i < 0$, то $C_i = 0$ та відновлення залишку

$$A_{i-1} = A_i + B \cdot 2^{-i} \ , \tag{4.11}$$

який приймається за залишок A_i' та процес продовжується згідно з формулою (4.11). Отже, операція ділення зводиться до послідовного виконання віднімань (додавань) у суматорі та зсувами дільника. Зсув дільника може бути замінений зсувом вмісту суматора в протилежний бік.

Алгоритм ділення, заснований на реалізації формул (4.10) та (4.11), називається *діленням з відновленням залишку*.

Розглянемо процес відновлення залишку. Коли $A_i < 0$ та $C_i = 0$, то на наступному кроці виконуються дії за формулою $A_{i+1} = A_i' - B \cdot 2^{-(i+1)} = A_i + B \cdot 2^{-i} - B \cdot 2^{-(i+1)}$. Після перетворення отримаємо

$$A_{i+1} = A_i + B \cdot 2^{-(i+1)}. \tag{4.12}$$

Таким чином, з'являється можливість побудувати алгоритм ділення за схемою $A_i = A_{i-1} - B \cdot 2^{-i}$, якщо $A_i \geq 0$, то $C_i = 1$ та перехід до наступного кроку; якщо $A_i < 0$, то $C_i = 0$ та продовження за формулою $A_{i+1} = A_i + B \cdot 2^{-(i+1)}$. При цьому, якщо $A_{i+1} \geq 0$, то $C_{i+1} = 1$ та переходимо до формули (4.10), якщо $A_{i+1} < 0$, то $C_{i+1} = 0$ та переходимо до формули (4.12). Такий алгоритм називається *діленням без відновлення залишків*.

Ділення двійкових чисел, які подано у формі з фіксованою комою на суматорах додаткового коду. При діленні знакова та цифрова частини частки отримуються окремо. Ділення чисел відбувається в три етапи:

1. Знаходиться знак частки за формулою

$$Sg_C = Sg_A \oplus Sg_B .$$

2. Проводиться нульовий крок ділення для перевірки частки на переповнення розрядної сітки шляхом алгебраїчного додавання діленого з дільником, якому приписується знак, протилежний знаку дільника.

3. Для знаходження цифр частки використовують співвідношення згідно з таблицею 4.1 (символ «-» над B вказує на зміну знака на протилежний на

наступному кроці ділення). В кінці ділення в $(n+1)$-й розряд частки обов'язково додається одиниця.

За своїм характером операція ділення відноситься до операцій, що дає не завжди точний результат, тому ознакою завершення операції ділення може бути досягнення заданої точності. Якщо в процесі ділення отримали залишок $A_i = 0$, то операція зупиняється та в розряди частки, що залишилися, записуються нулі. Зазвичай формальною ознакою завершення операції ділення є кількість зсувів: при досягненні кількості зсувів, що дорівнює кількості розрядів частки, в цифрових автоматах генерується сигнал завершення операції ділення.

Таблиця 4.1. Співвідношення для знаходження цифр частки

Знак діленого	+	+	-	-
Знак дільника	+	-	+	-
Операція в суматорі	$A_i + \overline{B} \cdot 2^{-i}$	$A_i + B \cdot 2^{-i}$	$A_i + B \cdot 2^{-i}$	$A_i + \overline{B} \cdot 2^{-i}$

Особливості ділення чисел, які подано у формі з плаваючою комою. Для отримання частки від ділення двох чисел, які подано у формі з плаваючою комою, необхідно визначити $m_C = m_A / m_B$ та $p_C = p_A - p_B$.

Оскільки мантиси діленого та дільника – нормалізовані числа, то при діленні можливі випадки, коли $|m_A| \geq |m_B|$ та $|m_A| < |m_B|$.

Якщо мантиса діленого більше або дорівнює мантисі дільника, то у кінці операції ділення знадобиться нормалізація частки (порушення правої границі). Таким чином, алгоритм ділення розпочинається з операції віднімання дільника від діленого та запису одиниці в цілу частину частки. Всі інші дії над мантисами аналогічні діям над числами, які подані у формі з фіксованою комою.

Якщо мантиса діленого менше мантиси дільника, то після операції віднімання на першому кроці отримується від'ємний залишок, що означає нуль в цілій частині мантиси частки та продовження алгоритму ділення за розглянутими вище правилами для чисел, які подані у формі з фіксованою комою. Таким чином, частка завжди виходить у прямому коді, а дії над мантисами здійснюються на ДСЗК або ДСДК.

ЛЕКЦІЯ 6

АРИФМЕТИЧНІ ОПЕРАЦІЇ У ДВІЙКОВО-ДЕСЯТКОВІЙ СИСТЕМІ

Д-код (двійково-кодоване подання) десяткового числа – таке його зображення, в якому кожна десяткова цифра зображується тетрадою з двійкових символів

$$A_Д = \left\{\alpha_4^1\alpha_3^1\alpha_2^1\alpha_1^1\right\}_1\left\{\alpha_4^2\alpha_3^2\alpha_2^2\alpha_1^2\right\}_2\ldots\left\{\alpha_4^n\alpha_3^n\alpha_2^n\alpha_1^n\right\}_n, \qquad (6.1)$$

де α_i^j – двійкові розряди тетради j; n – кількість десяткових розрядів.

Кількість різних Д-кодів визначається кількістю можливих поєднань по 10 з 16 комбінацій, які допускає тетрада. При утворенні Д-коду слід виходити із загальних вимог, що висуваються до систем числення:

1. різним десятковим цифрам повинні відповідати різні тетради;
2. більша десяткова цифра повинна зображуватися більшою тетрадою (якщо розряди тетради мають вагу по двійковій системі числення);
3. для десяткових цифр $a = \{\alpha_4\alpha_3\alpha_2\alpha_1\}$ та $b = \{\beta_4\beta_3\beta_2\beta_1\}$, що пов'язані співвідношенням $a + b = 9$, повинна задовольнятися умова

$$\beta_i = \begin{cases} 0, & якщо \quad \alpha_i = 1 \\ 1, & якщо \quad \alpha_i = 0 \end{cases} (i = 1, 2, 3, 4). \qquad (6.2)$$

Для однозначності переведення чисел в Д-код та навпаки бажано, щоб розряди тетрад мали певну вагу. Тоді значення десяткової цифри α_i, відповідає виразу

$$a_i = \alpha_4\sigma_4 + \alpha_3\sigma_3 + \alpha_2\sigma_2 + \alpha_1\sigma_1, \qquad (6.3)$$

де σ_i – вага розряду тетради.

У таблиці 6.1 представлено кодування десяткових цифр у різних Д-кодах.

Таблиця 6.1. Кодування десяткових цифр у різних Д-кодах

Десяткові числа	Еквіваленти в кодах						
	$Д_1$ (система 8421)	$Д_2$ (система 2421)	$Д_3$ (система 5121)	$Д_4$ (система 8421+3)	$Д_5$ (система 53-21)	$Д_6$ (система 75-31)	$Д_7$ (система 5421)
0	0000	0000	0000	0011	0000	0000	0000
1	0001	0001	0001	0100	0001	0001	0001
2	0010	0010	0010	0101	0111	0110	0010
3	0011	0011	0011	0110	1010	0111	0011
4	0100	0100	0111	0111	0101	1010	0100
5	0101	1011	1000	1000	1000	0100	1000
6	0110	1100	1001	1001	1001	0101	1001
7	0111	1101	1010	1010	1111	1000	1010
8	1000	1110	1011	1011	1100	1001	1011
9	1001	1111	1111	1100	1101	1110	1100

У таблиці для кожного Д-коду вказані дозволені комбінації. Всі інші комбінації – заборонені. Наявність дозволених та заборонених комбінацій –

дуже важлива властивість Д-кодів. Вона відрізняє їх від звичайних позиційних систем числення, в яких усі комбінації дозволені.

Розглянемо найпоширеніші Д-коди.

Код $Д_1$ прямого заміщення (система 8421 – ваги розрядів тетради відповідно дорівнюють 8, 4, 2, 1). У коді $Д_1$ дозволені комбінації відповідають двійковим еквівалентам десяткових цифр з вагами розрядів, що дорівнюють степеням основи 2. Для цього коду не виконується умова (5.2), оскільки цифри, що є доповненням до 9, не виходять простим інвертуванням наборів тетрад.

Для коду $Д_2$ (система 2421) ваги розрядів тетради відповідно дорівнюють 2, 4, 2, 1; таблиця кодування поділяється на дві частини: від 0 до 4 – тетради повторюють двійкові еквіваленти; від 5 до 9 – у порівнянні з двійковою системою кожна тетрада містить надлишок +0110. Це дає можливість будь-яку цифру однієї частини таблиці перетворити на її доповнення до 9 простим інвертуванням.

Для коду $Д_4$ (система 8421+3) усі тетради мають значення на три одиниці більше, ніж тетради коду $Д_1$ (звідси назва коду), та для нього не існує цілочисельних значень ваги, які задовольняли б вираз (5.3).

Коди $Д_5$ (система 53-21) та $Д_6$ (система 75-31) відрізняються від вищеназваних кодів тим, що для них деякі ваги мають від'ємні значення. В обчислювальних машинах різного призначення переважно використовуються коди $Д_1$ та $Д_4$.

Формальні правила порозрядного додавання десяткових чисел у Д-кодах. Для задавання формальних правил порозрядного додавання чисел, поданих в Д-коді, розглянемо ті особливості, які властиві цим кодам:

1. Наявність дозволених та заборонених комбінацій. Поява забороненої комбінації при виконанні будь-яких дій над числами свідчить про виникнення помилки або ж про необхідність ввести коригування результату.

2. При додаванні тетрад виникає потетрадне перенесення $П_i' = 16$ замість порозрядного перенесення $П_i' = 10$. Це призводить до необхідності корекції результату.

Насправді, якщо додаються числа $A_Д = a_n a_{n-1} \ldots a_1 a_0$ та $B_Д = b_n b_{n-1} \ldots b_1 b_0$, то сума $C_Д = A_Д + B_Д$ та

$$c_i = a_i + b_i + П_{i-1} - П_i q, \qquad (6.4)$$

де c_i – i-й розряд суми; $П_{i-1}$ – перенесення з молодшої тетради; $П_i$ – перенесення в старшу тетраду; $П_{i-1} = \{0,1\}$, $П_i = [0,1]$, $q = 10$.

Далі виведемо правила додавання стосовно Д-кодів.

При додаванні чисел в коді $Д_1$ можуть виникнути такі випадки:

1. Нехай $a_i + b_i + П_{i-1} < 10$, де a_i, b_i – тетради коду $Д_1$. При додаванні в даному розряді числа утворюється сума менше 10. Якщо дії над розрядами тетради проводяться за правилами двійкової арифметики, то правильний результат отримують без корекції.

2. Нехай $a_i + b_i + П_{i-1} \geq 10$, тобто виникає десяткове перенесення та сума дорівнює $a_i + b_i + П_{i-1} - П_i \cdot 10$, де $П_i = 1$.

Показником того, що результат невірний, є або поява забороненої комбінації, якщо $15 \geq a_i + b_i + П_{i-1} \geq 10$, або поява потетрадного перенесення $П'_i = 16$, що перевищує значення десяткового перенесення на 6. Отже, необхідна корекція результату в даній тетраді введенням поправки, що дорівнює +0110.

Правила потетрадного додавання чисел в $Д_1$-коді формулюються так:

- якщо при потетрадному додаванні перенесення в сусідню старшу тетраду не виникає ($П_i = 0$), то результат підсумовування не вимагає корекції;

- корекція результату потетрадного сумування, шляхом додавання поправки +0110 вимагається у разі, якщо:

а) виникає потетрадне перенесення в старшу тетраду ($П_i = 1$);

б) виникає заборонена комбінація.

Пристрій, який працює за сформульованими вище правилами, називається *однорозрядним десятковим суматором* для Д-коду.

При додаванні чисел у коді $Д_2$ можуть виникнути такі випадки:

1. Нехай $a'_i < 5$ та $b'_i < 5$, де a'_i, b'_i – тетради для коду $Д_2$. Тоді якщо $a'_i + b'_i + П_{i-1}$ – результат додавання не вимагає корекції; якщо $a'_i + b'_i + П_{i-1} \geq 5$ – результат попадає в другу частину таблиці кодування, де $c'_i = c_i + 6$.

Тут необхідна корекція результату, шляхом введення поправки 0110. Ознака цього – поява заборонених комбінацій.

2. Нехай $a'_i \geq 5$ та $b'_i < 5$; $15 > a'_i + b'_i + П_{i-1} \geq 5$.

Оскільки $a'_i = a_i + 6$, то при додаванні поправка не вимагається.

3. Нехай $a'_i \geq 5$ та $b'_i \geq 5$, $П_i = 1$. Тоді якщо $10 \leq a'_i + b'_i + П_{i-1} \leq 15$ – результат вимагає корекції шляхом введення поправки -0110, оскільки з'являється заборонена комбінація (виняток з цього випадку виникає при $a'_i + b'_i + П_{i-1} = 15$, коли немає необхідності вводити поправку, оскільки з'явиться дозволена комбінація); якщо $a'_i + b'_i + П_{i-1} < 15$ – результат не вимагає корекції.

При потетрадному додаванні в $Д_2$-коді результат додавання без корекції виходить в усіх випадках, окрім наступних: якщо за відсутності перенесення в старшу тетраду ($П_i = 0$) виникає заборонена комбінація, то необхідно ввести поправку +0110; якщо за наявності перенесення в старшу тетраду ($П_i = 1$) виникає заборонена комбінація, то вимагається ввести поправку -0110 (1001 в зворотному коді або 1010 – в додатковому коді).

Оскільки поправки бувають додатні або від'ємні, то їх введення супроводжується блокуванням ланцюгів міжтетрадного перенесення в період корекції результату.

При додаванні чисел в коді $Д_4$ можливі наступні випадки. Нехай $a''_i = a_i + 3$, $b''_i = b_i + 3$, де a''_i та b''_i – тетради для коду $Д_4$. Тоді: якщо $a''_i + b''_i + П_{i-1} < 10$, то $c''_i = a_i + 3 + b_1 + 3 + П_{i-1} = \underbrace{a_i + b_i + П_{i-1} + 3}_{c_i} + 3$; результат

потребує корекції шляхом введення поправки -0011; якщо $a_i'' + b_i'' + \Pi_{i-1} \geq 10$, то $c_i'' = a_i + b_i + \Pi_{i-1} + 3 + 3$.

Тут виникає десяткове перенесення, яке, за умовою, "відносить" з собою шість надмірних комбінацій. Отже, в даному випадку потрібно провести корекцію результату шляхом введення поправки $+0011$.

Тут поправки вводяться при блокуванні ланцюгів потетрадного перенесення.

Правила введення поправок можна сформулювати так: якщо при додаванні тетрад не виникає перенесення $(\Pi_i = 0)$, то поправка дорівнює -0011 (чи доповнення $+1101$); якщо ж виникає потетрадне перенесення $(\Pi_i = 1)$, то поправка дорівнює $+0011$.

Аналогічним чином можна розглянути правила додавання для інших Д-кодів.

Подання від'ємних чисел у Д-кодах. Подавання Д-коду в розрядній сітці машини може здійснюватися у формі з фіксованою або плаваючою комою. При цьому від'ємні числа повинні бути подані в прямому, зворотному або додатковому кодах. Тому, якщо $A = -0, a_1 a_2 \ldots a_n$, де a_i – тетради, то

$$\begin{aligned} [A]_{пр} &= 1, a_1 a_2 \ldots a_n; \\ [A]_{зв} &= 1, \bar{a}_1 \bar{a}_2 \ldots \bar{a}_n; \\ [A]_{д} &= 1, \tilde{a}_1 \tilde{a}_2 \ldots \tilde{a}_n, \end{aligned} \qquad (6.5)$$

де $\bar{a}_i$ – доповнення до $q-1$ в усіх тетрадах; $\tilde{a}_i$ – доповнення до $q-1$ в усіх тетрадах, за винятком молодшої, для якої це доповнення до $q = 10$.

З правил перетворення (5.5) виходить, що

$$a_i + a_i = q - 1. \qquad (6.6)$$

Це означає, що для Д-кодів, для яких виконується умова (6.6), зворотний код виходить простим інвертуванням набору тетрад.

Код $Д_1$ відрізняється тим, що для нього не виконується умова (6.2). Ця особливість коду впливає на утворення зворотного або додаткового коду, оскільки інвертування набору тетрад означає отримання доповнення до $2^4 - 1 = 15$. Отже, необхідно забрати різницю. Один з прийомів, який використовується при цьому, полягає в тому, що до всіх цифрових тетрад числа в коді $Д_1$ додається $+0110$ та після цього робиться інвертування набору. Отримане зображення є зворотним кодом числа.

Множення чисел у Д-кодах. Виконання операцій множення в Д-кодах проводиться за класичною схемою: множення чисел зводиться до послідовного підсумовування частинних добутків, отриманих у результаті добутку множеного на чергову цифру множника. Оскільки кожна цифра множника подається у вигляді $(\beta_4 \beta_3 \beta_2 \beta_1)_i$, де i – номер розряду, то множення супроводжується розшифруванням значення чергової тетради множника та зсувом на чотири розряди відразу. Розшифрування можна здійснити різними способами. Простим прийомом є послідовне віднімання одиниці зі значення тетради до отримання нуля та відповідно додавання множеного в суматор на

кожному такті. Оскільки при множенні множеного на тетраду можливе переповнення розрядної сітки суматора (внаслідок того, що множене додається до суми частинних добутків стільки разів, скільки одиниць знаходиться в даному десятковому розряді множника), то в ньому необхідно передбачити додаткову тетраду для перенесень. З чотирьох можливих методів множення в Д-кодах доцільно використовувати тільки один: множення молодших розрядів множника із зсувом суматора частинних добутків вправо. Множення переважно проводиться в прямому коді. Тобто знак результату визначається сумою цифр множників за модулем 2.

Ділення чисел у Д-кодах. Ділення десяткових чисел у Д-кодах виконується методом послідовного віднімання дільника з діленого на першому кроці та із залишків – на подальших кроках. Віднімання на кожному кроці робиться до тих пір, поки не вийде від'ємний залишок. Кожного разу при отриманні додатного залишку додається одиниця в спеціальний лічильник, де накопичується чергова цифра частки. Потім здійснюється зсув на чотири двійкові розряди та додавання дільника до тих пір, поки не отримаємо додатний залишок. Кількість додавань (без останнього) є доповненням відповідної цифри частки до 9, що заноситься в лічильник чергової цифри частки.

Таким чином, процес ділення складається з ряду послідовних циклів додавання та віднімання з зсувом. Знак частки виходить як логічна сума за модулем 2 знаків чисел.

Усі дії при виконанні операції ділення повинні здійснюватися на суматорі додаткового (зворотного) коду, який працює за правилами множення – віднімання відповідного Д-коду.

ЛЕКЦІЯ 8

ПРИНЦИП МІКРОПРОГРАМНОГО КЕРУВАННЯ. ПОНЯТТЯ ОПЕРАЦІЙНОГО ТА КЕРУЮЧОГО АВТОМАТІВ

При описуванні функціонування різних засобів обчислювальної техніки використовується їх подання у вигляді моделі В.Н. Глушкова – сукупності *керуючого* та *операційного автоматів* (рисунок 8.1).

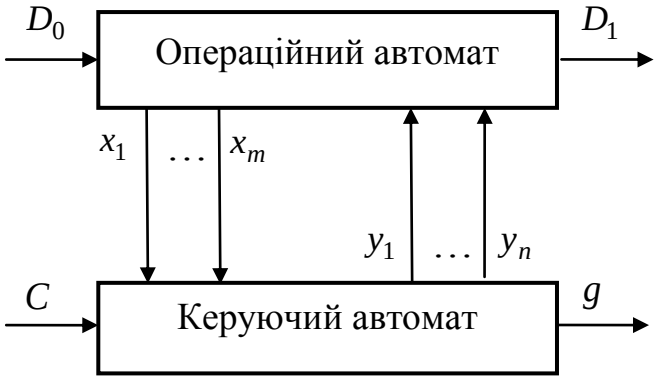


Рисунок 8.1. Модель цифрового пристрою

Операційний автомат (ОА) служить для зберігання слів інформації, виконання набору мікрооперацій та обчислення значень логічних умов, тобто операційний автомат є структурою, що організована для виконання дій над інформацією. На вхід ОА подаються вхідні дані D_0, які відповідно до алгоритму операції перетворюються у вихідні дані D_1. ОА також генерує множину $\{x_1,\dots x_m\}$ інформаційних сигналів (логічних умов) для керуючого автомата.

Керуючий автомат (КА) генерує послідовність керуючих сигналів $\{y_1,\dots y_n\}$, що забезпечують виконання в операційному автоматі задану послідовність елементарних дій, яка реалізує алгоритм виконання операції. Керуюча послідовність генерується відповідно до заданого алгоритму та з врахуванням значень логічних умов $\{x_1,\dots x_m\}$, що формуються ОА.

Переважно операційний пристрій може виконувати декілька різних операцій (наприклад, арифметико-логічний пристрій процесора може виконувати чотири арифметичні дії та декілька логічних операцій над вхідними словами). В цьому випадку на вхід КА надходить команда C, що визначає тип виконуваної операції. Крім того, оскільки різні операції над різними даними виконуються за різний час, КА формує сигнал g, що відмічає закінчення операції та готовність вихідних даних.

Отже, будь-який операційний пристрій – процесор (який зазвичай, у свою чергу, представляють тим, що складається з двох операційних пристроїв: арифметико-логічного пристрою та центрального пристрою керування), канал вводу/виводу, контролер зовнішнього пристрою – можна представити як

композицію операційного та керуючого автоматів. Операційний автомат, реалізовуючи дії над словами інформації, є виконавчою частиною пристрою, роботу якого організовує керуючий автомат, що генерує необхідні послідовності керуючих сигналів.

Такий підхід дає змогу розробити ефективні процедури синтезу ОА та КА, формалізувати ці процедури та, в деяких випадках, автоматизувати процес синтезу цифрових пристроїв.

Функціональна та структурна організація операційних пристроїв базується на *принципі мікропрограмного керування*, який полягає в наступному:

1. Елементарна неподільна дія опрацювання інформації в операційному автоматі, що відбувається протягом одного моменту автоматного часу (одного такту роботи автомата), називається *мікрооперацією*. Множину мікрооперацій, що реалізується в операційному автоматі, позначимо як $Y = \{y_1, \dots y_n\}$.

2. Якщо в операційному автоматі одночасно реалізується кілька мікрооперацій, то така множина мікрооперацій називається *мікрокомандою*. Так, якщо $Y_t = \{y_{t1}, \dots y_{tU_t}\}$ – мікрокоманда, мікрооперації $y_{t1}, \dots y_{tU_t}$ виконуються в один й той самий момент автоматного часу ($y_{tu} \in Y, \quad u = \overline{1, U_t}$). При $U_t = 1$ мікрокоманда Y_t складається з однієї мікрооперації. У випадку $U_t = 0$ – множина мікрооперацій, що утворює мікрокоманду, є пустою множиною. Мікрокоманду, що складається з пустої множини мікрооперацій, позначають через Y_0. Реалізація такої мікрокоманди рівносильна відсутності виконання будь-яких елементарних операцій.

3. Нехай $Y_1, \dots, Y_T$ – мікрокоманди, які можуть виконуватися в операційному автоматі. Послідовність виконання мікрокоманд визначається *функціями переходу* – булевими функціями α_{ij}, $i, j = \overline{1, T}$ від множини двійкових змінних $X = \{x_1, \dots x_m\}$ – вхідних змінних керуючого автомату. З кожною мікрокомандою Y_i пов'язана множина функцій переходу $\{\alpha_{i1}, \dots, \alpha_{iT}\}$, така, що якщо після виконання мікрокоманди $Y_i \alpha_{it} = 1$, то наступною буде виконуватися мікрокоманда Y_i. Множина функцій переходу з однієї і тої самої мікрокоманди Y_i має властивості *ортогональності* ($\alpha_{ij}\alpha_{it} = 0$ при $j \neq t$) та *повноти* ($\bigcup_{j=1}^{T} \alpha_{ij} = 1$). Ортогональність говорить про те, що безпосередньо після даної мікрокоманди при визначених значеннях вхідних змінних може виконуватися тільки одна мікрокоманда, а повнота – що вона виконається обов'язково.

4. Процес виконання операцій в пристрої описується у формі алгоритму, який подається в термінах мікрооперацій та логічних умов та називається *мікропрограмою*. Мікропрограма визначає порядок перевірки значень логічних умов та проходження мікрооперацій, необхідний для отримання потрібних результатів.

5. Мікропрограма використовується як форма представлення функції пристрою, на основі якої визначається структура та порядок функціонування пристрою в часі.

ЛЕКЦІЯ 9

АБСТРАКТНИЙ СИНТЕЗ КЕРУЮЧИХ АВТОМАТІВ З ЖОРСТКОЮ ЛОГІКОЮ ПО ГРАФ-СХЕМІ АЛГОРИТМУ

Скінченний керуючий автомат, що реалізує мікропрограму роботи дискретного пристрою, прийнято називати *мікропрограмним автоматом*. Як вже зазначалося, мікропрограма відображається за допомогою ГСА. Розглянемо послідовність етапів синтезу керуючого автомата за його ГСА:

1. Побудова змістовної ГСА.
2. Побудова маркованої ГСА.
3. Побудова графа переходів автомата.
4. Проведення структурного синтезу автомата за його графом переходів відомими методами, наприклад, за допомогою канонічного методу структурного синтезу.

Побудова маркованої ГСА проводиться за змістовною ГСА. Для автоматів Мура та Мілі процедура маркування відрізняється.

Якщо необхідно побудувати мікропрограмний автомат Мілі, то змістовна ГСА керуючого автомата маркується відповідно до наступних правил:

1) символом стану a_1 маркується вхід вершини, що йде за вершиною «початок» та вхід вершини «кінець»;
2) входи всіх вершин, що йдуть за операторними, мають бути марковані символами a з індексом;
3) якщо вихід вершини маркується, то тільки одним символом;
4) входи різних вершин, за винятком вершини «кінець», маркуються різними символами;
5) змістовні терміни мікрооперацій та логічних умов замінюються їх умовними позначеннями.

Приклад маркованої ГСА мікропрограмного автомата Мілі подано на рисунку 9.1.

Якщо необхідно побудувати мікропрограмний автомат Мура, то змістовна ГСА керуючого автомата маркується відповідно до наступних правил:

1) символом a_1 маркуються вершини «початок» та «кінець»;
2) різні операторні вершини маркуються різними символами;
3) всі операторні вершини мають бути промарковані.

Приклад маркованої ГСА мікропрограмного автомата Мура поданий на рисунку 9.2.

Після отримання маркованої ГСА будується граф переходів автомата. Він має стільки різних вершин, скільки різних букв a з індексами знаходиться у маркованій ГСА. Кожна вершина графу переходів автомата маркується буквою a з відповідним індексом. Між двома вершинами a_i та a_j графу знаходиться дуга, якщо на маркованій ГСА між вершинами з позначками a_i та a_j є шлях.

Над дугою ставиться вхідний сигнал, що дорівнює кон'юнкції логічних умов відповідного шляху в маркованій ГСА. При цьому виконанню логічної умови відповідає змінна без заперечення, а невиконанню логічної умови – змінна із запереченням на відповідній дузі графу переходів автомата. Якщо у маркованій ГСА між згаданими вершинами з позначками a_i та a_j існує кілька шляхів, то в графі переходів автомата на дузі, що зв'язує a_i та a_j, через символ диз'юнкції записуються всі кон'юнкції, які відповідають існуючим шляхам. Якщо будується граф переходів автомата Мура, то символи мікрооперацій (вихідні сигнали керуючого автомата) записуються біля відповідних його вершин. Для автомата Мілі символи мікрооперацій записуються на відповідних дугах при кон'юнкціях логічних умов, які описують шлях через операторну вершину з мікрооперацією, що розглядається. Якщо у маркованій ГСА існує безумовний перехід між операторними вершинами, тобто шлях, який не проходить ні через будь-які умовні вершини, то на графі переходів автомата йому відповідає дуга, якій приписується вхідний сигнал «1», який вказує, що даний перехід в автоматі здійснюється при надходженні наступного синхросигналу. Графи переходів автоматів Мілі та Мура для маркованих ГСА (рисунки 9.1 та 9.2) зображено на рисунках 9.3 та 9.4 відповідно.

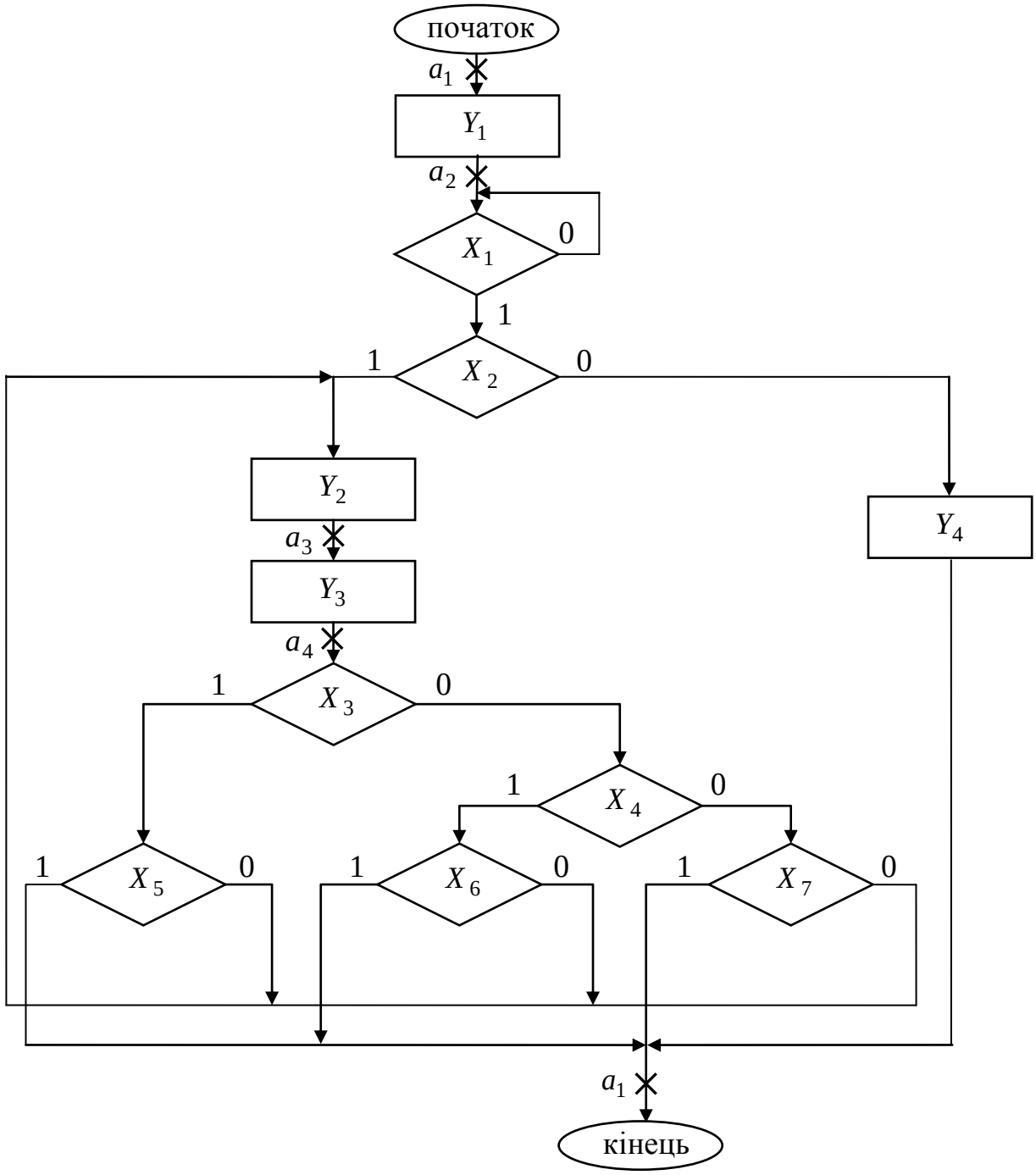


Рисунок 9.1. Маркована ГСА автомата Мілі

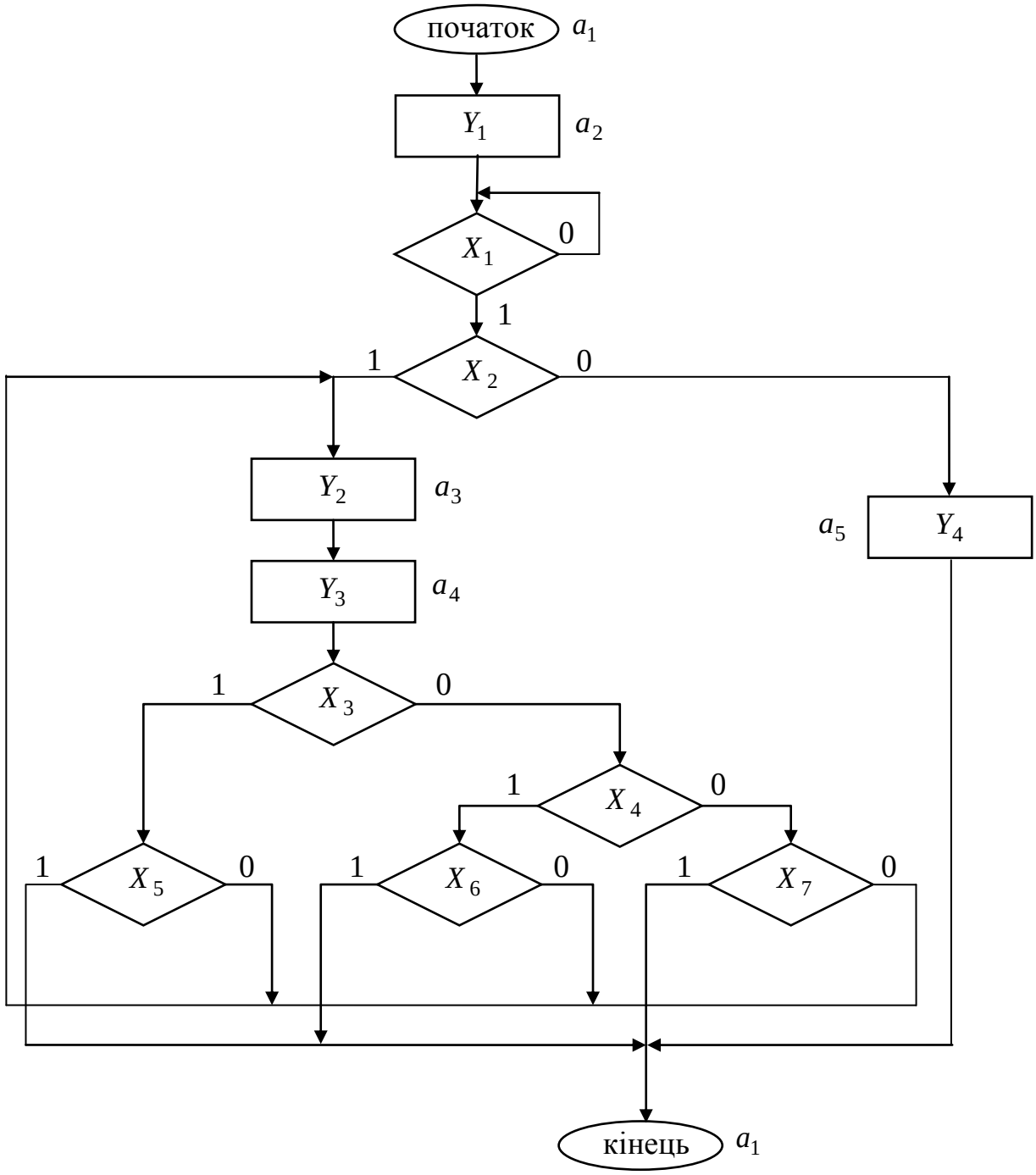
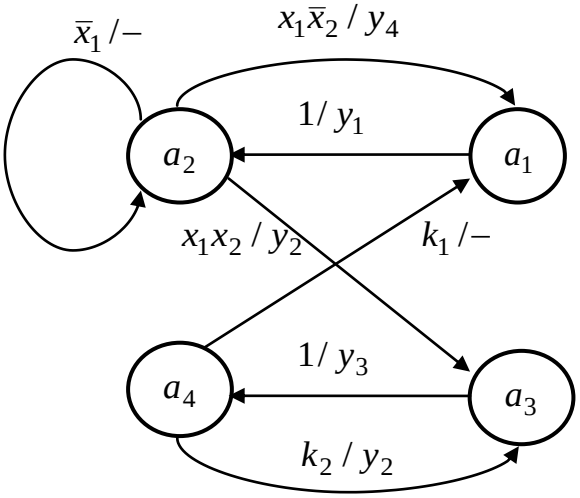
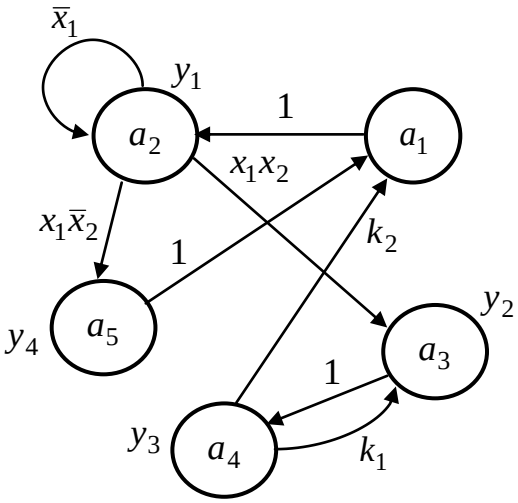


Рисунок 9.2. Маркована ГСА автомата Мура



$$k_1 = x_3x_5 + \bar{x}_3x_4x_6 + \bar{x}_3\bar{x}_4x_7,$$
$$k_2 = x_3\bar{x}_5 + \bar{x}_3x_4\bar{x}_6 + \bar{x}_3\bar{x}_4\bar{x}_7$$

Рисунок 9.3. Граф автомата Мілі



$$k_1 = x_3\bar{x}_5 + \bar{x}_3x_4\bar{x}_6 + \bar{x}_3\bar{x}_4\bar{x}_7,$$
$$k_2 = x_3x_5 + \bar{x}_3x_4x_6 + \bar{x}_3\bar{x}_4x_7$$

Рисунок 9.4. Граф автомата Мура

ЛЕКЦІЯ 10

СТРУКТУРНИЙ СИНТЕЗ КЕРУЮЧИХ АВТОМАТІВ З ЖОРСТКОЮ ЛОГІКОЮ

Структурний синтез мікропрограмних автоматів після отримання графу аналогічний канонічному методу синтезу цифрових автоматів, розглянутому раніше. Проте існують й певні особливості, в першу чергу пов'язані з тим, що для реальних автоматів кількість елементів пам'яті та вхідних сигналів може сягати десяти й більше. Функції збудження та вихідні сигнали важко піддаються мінімізації та й мінімізація не дає істотного спрощення цих функцій при великій кількості змінних. Тому мінімізація практично не використовується при синтезі мікропрограмних автоматів.

При виконанні структурного синтезу будують так звані структурні таблиці переходів-виходів, які можуть бути як прямими, так і зворотними.

Розглянемо етапи структурного синтезу на конкретних прикладах.

Структурний синтез автомата Мілі. Використовуючи канонічний метод структурного синтезу, побудуємо функціональну схему автомата Мілі, граф якого зображено на рисунку 9.3. Таблиця переходів-виходів автомата, що відповідає графу з рисунку 9.3, подана в таблиці 10.4. Кодування вхідних сигналів може бути здійснено, наприклад, відповідно до таблиць 10.5, 10.6 та 10.7.

Таблиця 10.4. Таблиця переходів-виходів автомата Мілі

Стани автомата	Вхідні сигнали					
	1	$\bar{x}_1$	$x_1\bar{x}_2$	x_1x_2	k_1	k_2
a_1	a_2 / y_1	-	-	-	-	-
a_2	-	$a_2 / -$	a_1 / y_4	a_3 / y_2	-	-
a_3	a_4 / y_3	-	-	-	-	-
a_4	-	-	-	-	$a_1 / -$	a_3 / y_2

Таблиця 10.5. Кодування вхідних сигналів

Вхідні сигнали	Код вхідних сигналів
$\bar{x}_1$	001
$x_1\bar{x}_2$	010
x_1x_2	011
k_1	100
k_2	101

Таблиця 10.6. Кодування станів автомата

Стани автомата	Код стану
a_1	00
a_2	01
a_3	10
a_4	11

Таблиця 10.7. Кодування вихідних сигналів

Вихідні сигнали	Код вихідних сигналів
y_1	00
y_2	01
y_3	10
y_4	11

На практиці при синтезі керуючого автомата по ГСА етапом кодування вхідних сигналів (таблиця 10.5) можна знехтувати та будувати КС збудження та

виходів автомата, використовуючи тільки схему формування його вхідних сигналів. Для вихідних сигналів автомата y_1, y_2, y_3, y_4 часто зручніше використовувати окремі фізичні шини (для кожного вихідного сигналу свою фізичну шину). Отже, кожній логічній умові ГСА поставимо у відповідність фізичну одно розрядну шину, а кожен вихідний сигнал автомата також реалізуємо окремою фізичною шиною.

Закодуємо таблицю переходів-виходів автомата Мілі (таблиця 10.4) згідно з закодованими станами автомата (таблиця 10.6). Закодована таблиця переходів-виходів відповідає таблиці 10.8.

Таблиця 10.8. Закодована таблиця переходів-виходів автомата Мілі

Стани автомата	Вхідні сигнали					
Q_1Q_2	1	$\bar{x}_1$	$x_1\bar{x}_2$	x_1x_2	k_1	k_2
00	$01/y_1$	-	-	-	-	-
01	-	$01/-$	$00/y_4$	$10/y_2$	-	-
10	$11/y_3$	-	-	-	-	-
11	-	-	-	-	$00/-$	$10/y_2$

Як елементи пам’яті автомата виберемо *T*-тригери (їх буде два, оскільки кількість станів дорівнює чотирьом). Тоді таблиця функцій збудження синтезованого автомата буде мати вигляд згідно з таблицею 10.9, а функції збудження елементів пам’яті T_1 та T_2, отримані за цією таблицею, запишуться так:

$$T_1 = \overline{Q}_1Q_2x_1x_2 + Q_1Q_2k_1 = \overline{Q}_1Q_2x_1x_2 + Q_1Q_2\left(x_3\bar{x}_5 + \bar{x}_3x_4\bar{x}_6 + \bar{x}_3\bar{x}_4\bar{x}_7\right),$$

$$T_2 = \overline{Q}_1\overline{Q}_2 + Q_1\overline{Q}_2 + \overline{Q}_1Q_2x_1\bar{x}_2 + \overline{Q}_1Q_2x_1x_2 + Q_1Q_2k_1 + Q_1Q_2k_2.$$

Після спрощення функція T_2 набуде такого вигляду:

$$T_2 = \overline{Q}_2 + \overline{Q}_1Q_2x_1 + Q_1Q_2.$$

Таблиця 10.9. Таблиця функцій збудження синтезованого автомата Мілі

Стани автомата	Вхідні сигнали					
Q_1Q_2	1	$\bar{x}_1$	$x_1\bar{x}_2$	x_1x_2	k_1	k_2
00	$01/y_1$	-	-	-	-	-
01	-	$00/-$	$01/y_4$	$11/y_2$	-	-
10	$01/y_3$	-	-	-	-	-
11	-	-	-	-	$11/-$	$01/y_2$

Функції виходів можуть бути отримані за таблицею 10.9 або бути подані співвідношеннями

$y_1 = \overline{Q}_1\overline{Q}_2$,

$$y_2 = \overline{Q}_1 Q_2 x_1 x_2 + Q_1 Q_2 k_2,$$
$$y_3 = Q_1 \overline{Q}_2,$$
$$y_4 = \overline{Q}_1 Q_2 x_1 \bar{x}_2.$$

Структурний синтез автомата Мура. Виконаємо структурний синтез керуючого автомата Мура згідно методики [1]. Автомат Мура (рисунок 9.4) можна задати прямою або зворотною таблицею переходів-виходів (таблиця 10.10 або 10.11). Як приклад, синтез виконуватимемо за прямою таблицею (таблиця 10.10).

В автоматі, що розглядається, кількість станів дорівнює п'яти, отже кількість елементів пам'яті дорівнюватиме трьом**.**

Таблиця 10.10. Пряма таблиця переходів автомата Мура

$a_i(y)$	a_j	x
$a_1(-)$	a_2	1
$a_2(y_1)$	a_2 a_3 a_5	$\bar{x}_1$ $x_1 x_2$ $x_1 \bar{x}_2$
$a_3(y_2)$	a_4	1
$a_4(y_3)$	a_3 a_1	k_1 k_2
$a_5(y_4)$	a_1	1

Таблиця 10.11. Зворотна таблиця переходів автомата Мура

a_j	$a_i(y)$	x
a_4 a_5	$a_1(-)$	k_2 1
a_1 a_2	$a_2(y_1)$	1 $\bar{x}_1$
a_2 a_4	$a_3(y_2)$	$x_1 x_2$ k_1
a_3	$a_4(y_3)$	1
a_2	$a_5(y_4)$	$x_1 \bar{x}_2$

Закодуємо внутрішні стани автомата, використовуючи при цьому карту Карно та метод сусіднього кодування (рисунок 10.21).

	00	01	11	10
0	a_1	a_2	a_3	a_4
1	a_5			

a_1=000 a_4=010
a_2=001 a_5=100
a_3=011

Рисунок 10.21. Карта Карно для кодування станів автомата Мура

Побудуємо структурну таблицю переходів-виходів автомата Мура (таблиця 10.12).

Для структурного синтезу автомата будемо використовувати JK-тригери. В даній таблиці у стовпцях $K(a_i)$ та $K(a_j)$ вказується код початкового стану та стану переходу відповідно. У стовпці функцій збудження (ФЗ) вказуються ті значення функцій збудження для JK-тригера, які на даному переході

обов'язково дорівнюють 1. Інші (тобто які дорівнюють 0 або ті, що приймають невизначені значення) не вказуються. Це еквівалентно тому, що всім невизначеним значенням функцій збудження приписується значення 0, що в загальному випадку не дає мінімізації функції, однак у реальних автоматах мінімізація звичайно не робиться з причини її неефективності.

Таблиця 10.12. Структурна таблиця переходів-виходів автомата Мура

$a_i(y)$	$K(a_i)$	a_j	$K(a_j)$	x	ФЗ
$a_1(-)$	000	a_2	001	1	J_3
$a_2(y_1)$	001	a_2 a_3 a_5	001 011 100	$\bar{x}_1$ x_1x_2 $x_1\bar{x}_2$	- J_2 J_1, K_3
$a_3(y_2)$	011	a_4	010	1	K_3
$a_4(y_3)$	010	a_3 a_1	011 000	k_1 k_2	J_3 K_2
$a_5(y_4)$	100	a_1	000	1	K_1

Вирази для функцій збудження JK-тригерів записуємо у вигляді логічної суми добутків $a_i x$, де a_i – початковий стан, x – умова переходу. Для спрощення отриманих виразів виконуємо всі можливі операції склеювання та поглинання:

$$J_1 = a_2 x_1 \bar{x}_2,$$

$$J_2 = a_2 x_1 x_2,$$

$$J_3 = a_1 + a_4 k_1 = a_1 + a_4(x_3\bar{x}_5 + \bar{x}_3 x_4 \bar{x}_6 + \bar{x}_3\bar{x}_4\bar{x}_7),$$

$$K_1 = a_5,$$

$$K_2 = a_4 k_2 = a_4(x_3 x_5 + \bar{x}_3 x_4 x_6 + \bar{x}_3\bar{x}_4 x_7),$$

$$K_3 = a_3 + a_2 x_1 \bar{x}_2.$$

Вирази для вихідних сигналів автомата Мура отримуємо, виходячи з того, що ці сигнали визначаються тільки внутрішнім станом автомата:

$$y_1 = a_2,$$

$$y_2 = a_3,$$

$$y_3 = a_4,$$

$$y_4 = a_5.$$

Для побудови функціональної схеми автомата Мура за отриманими виразами необхідно або замінити стани a_i його значеннями через Q_1, Q_2, Q_3 або отримати сигнал, відповідний a_i. Використаємо другий спосіб та для отримання сигналу a_i застосуємо дешифратор станів (дешифратор 3-8), на вхід якого надходять сигнали з виходів елементів пам'яті Q_1, Q_2, Q_3. Функціональна схема автомата Мура зображена на рисунку 10.22.

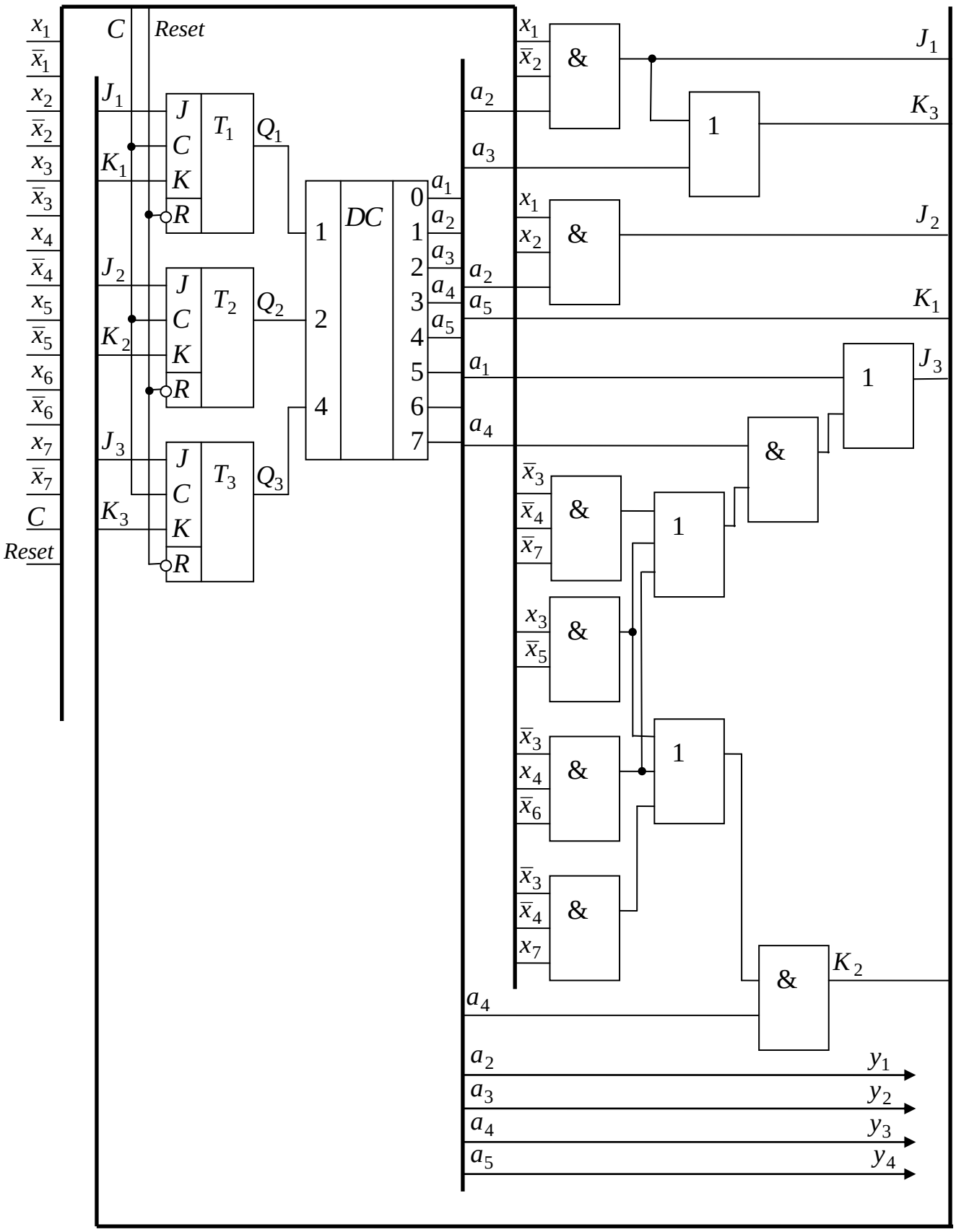


Рисунок 10.22. Функціональна схема мікропрограмного автомата Мура

ЛЕКЦІЯ 11

СИНТЕЗ КЕРУЮЧОГО АВТОМАТА З ПРОГРАМОВАНОЮ ЛОГІКОЮ

Принцип синтезу КАПЛ розглянемо на основі прикладу.

Приклад. Спроектувати КАПЛ за мікропрограмою, поданою у вигляді ГСА (рисунок 11.1).

Загальна структура такого пристрою подана на рисунку 11.2. Виходячи з можливих варіантів організації адресації та способів кодування поля мікрооперацій, виберемо *природну адресацію* та *змішаний спосіб кодування* мікрооперацій. Обмежимося єдиним форматом мікрокоманди.

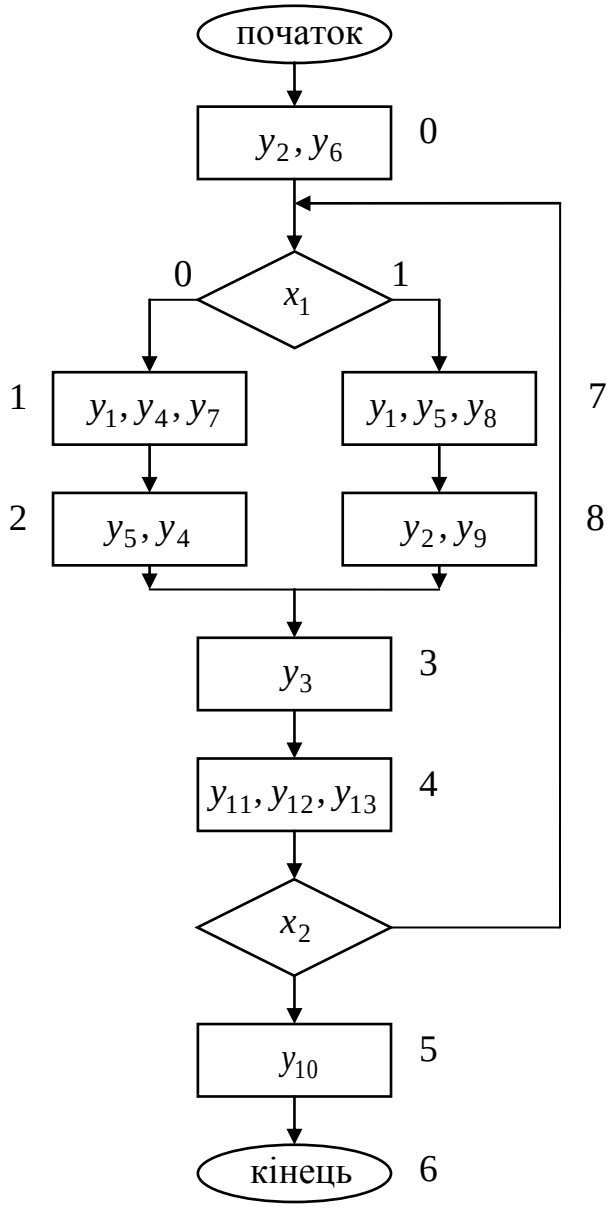


Рисунок 11.1. Початкова мікропрограма для проектування автомата

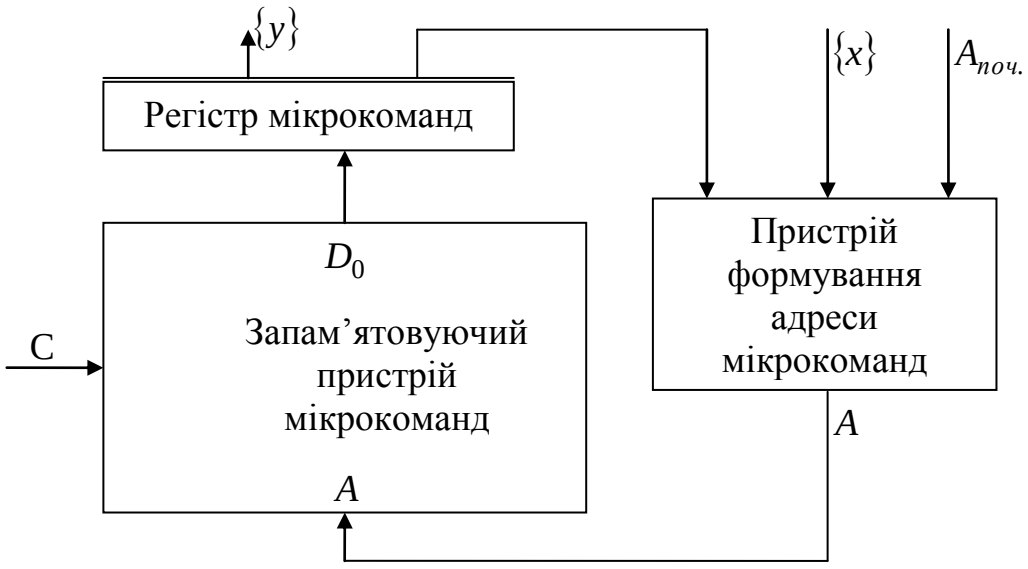


Рисунок 11.2. Мікропрограмний автомат з програмованою логікою

На розрядність полів мікрокоманди впливають наступні параметри:

- кількість різних мікрооперації, що формуються КА, визначає (з урахуванням вибраного способу кодування) довжину поля мікрооперації;
- кількість різних логічних умов визначає довжину поля x;
- кількість вершин ГСА пов'язана із загальною кількістю мікрокоманд, а отже, з об'ємом пам'яті мікропрограм та розрядністю поля адреси мікрокоманди.

Множина мікрооперацій Y, що використовуються в заданій ГСА – $Y = \{y_1, y_2, \ldots, y_{13}\}$, нараховує тринадцять елементів. При горизонтальному кодуванні поле мікрооперацій буде займати тринадцять розрядів. Вертикальний спосіб кодування мікрооперацій до заданої ГСА є непридатний, оскільки ГСА містить вершини з двома та трьома мікроопераціями.

Спробуємо реалізувати розбиття множини Y на підмножини несумісних мікрооперацій. Розбиваємо початкову множину на три підмножини Y_1, Y_2, Y_3 та розмістимо в них мікрооперації операторної вершини, що має s мікрооперацій. Якщо в ГСА таких вершин декілька – вибираємо будь-яку з них. В нашому випадку маємо:

$$Y_1 = \{y_1\},\ Y_2 = \{y_4\},\ Y_3 = \{y_7\}.$$

Далі розмістимо по множинам мікрооперацій такі вершини, що містять (в нашому випадку) три мікрооперації:

$$Y_1 = \{y_1\},\ Y_2 = \{y_4, y_5\},\ Y_3 = \{y_7, y_8\}.$$

Зауважимо, що перша мікрооперація другої вершини, що розглядається, співпадає з першою мікрооперацією першої вершини. Вона вже присутня в множині Y_1 ($y_1 \in Y_1$), тому не включається повторно. Нарешті, розмістимо мікрооперації третьої потрійної вершини:

$$Y_1 = \{y_1, y_{11}\},\ Y_2 = \{y_4, y_5, y_{12}\},\ Y_3 = \{y_7, y_8, y_{13}\}.$$

Тепер нерозподіленими залишилися мікрооперації (деякі) подвійних та одинарних вершин. Вершина (y_2, y_6) – обидві мікрооперації несумісні з вже розприділеними, тому можуть розташовуватися довільно, тільки щоб вони знаходилися в різних підмножинах:

$$Y_1 = \{y_1, y_{11}, y_2\},\ Y_2 = \{y_4, y_5, y_{12}, y_6\},\ Y_3 = \{y_7, y_8, y_{13}\}.$$

Вершина (y_2, y_9) – y_9 неможна розташовувати в Y_1 оскільки сумісна з нею $y_2 \in Y_1$. Підмножини краще заповнювати рівномірно, тому розмістимо y_9 в Y_3:

$$Y_1 = \{y_1, y_{11}, y_2\},\ Y_2 = \{y_4, y_5, y_{12}, y_6\},\ Y_3 = \{y_7, y_8, y_{13}, y_9\}.$$

Залишилося дві нерозподілені мікрооперації – y_3 та y_{10}, перша з яких сумісна з y_5, тому її неможна розміщувати в Y_2, а друга несумісна ні з ніякими іншими та може розміщуватися довільно. Розмістимо їх в множину, що має поки найменшу кількість елементів – в Y_1:

$$Y_1 = \{y_1, y_{11}, y_2, y_3, y_{10}\},\ Y_2 = \{y_4, y_5, y_{12}, y_6\},\ Y_3 = \{y_7, y_8, y_{13}, y_9\}.$$

Всі тринадцять мікрооперацій розподілились по трьом підмножинам. Але КА зазвичай має виробляти ще одну мікрооперацію, що вказує про завершення виконання алгоритму та призначену для використання не в ОА, а в КА верхнього рівня ієрархії. Назвемо цю мікрооперацію y_k та включимо в довільну множину (наприклад, в Y_2), оскільки вона, природно, несумісна з жодною із мікрооперацій. Отже, отримали наступний розподіл:

$$Y_1 = \{y_1, y_{11}, y_2, y_3, y_{10}\},\ Y_2 = \{y_4, y_5, y_{12}, y_6, y_k\},\ Y_3 = \{y_7, y_8, y_{13}, y_9\}.$$

Для кодування елементів кожного з трьох підмножин необхідно по три двійкових розряди. Треба зазначити, що в кожній підмножині необхідно передбачити один код на випадок відсутності мікрооперації з цієї підмножини в мікрокоманді.

В підмножині Y_3, всього одна "зайва" мікрооперація, а серед кодів Y_1 та Y_2 є вільні. Перенесемо одну із мікрооперацій з Y_3 в іншу підмножину, зберігаючи вимогу до попарної несумісності всіх мікрооперацій однієї підмножини. Очевидно, перші три елемента підмножини Y_3 неможливо перенести в іншу, оскільки вони є мікрооперaціями з "потрійних" вершин. Зате мікрооперація y_9 сумісна тільки з y_9, тому y_9 можна перенести в Y_2. Таким чином, попередньо впорядкувавши елементи підмножин в порядку зростання індексів, отримаємо:

$$Y_1 = \{y_1, y_2, y_3, y_{10}, y_{11}\},\ Y_2 = \{y_4, y_5, y_6, y_9, y_{12}, y_k\},\ Y_3 = \{y_7, y_8, y_{13}\}.$$

Тепер визначимо розміри полів мікрокоманди. Поле мікрооперації буде складатися з трьох підполів – Y_1, Y_2, Y_3 (назвемо їх за іменами відповідних підмножин), розміром в 3, 3 та 2 двійкових розряди відповідно.

Поле номеру умови x має містити номер однієї з двох логічних умов – x_1 або x_2, але для підвищення гнучкості процесу мікропрограмування зручно мати можливість вибирати ще й тотожно істинну та тотожно хибну умову. Отже, поле x буде займати два розряди.

Нарешті, поле адреси визначається об'ємом пам'яті мікропрограм. Якщо в нашому прикладі ми будемо вважати, що розробляємо КА тільки для реалізації мікропрограми з рисунку 11.1, а вона має 8 вершин, не рахуючи початкової, кінцевої та умовних, то кількість мікрокоманд (кожна мікрокоманда – комірка пам'яті, що має свою адресу), які видаються КА, буде не менше 8. Для поля адреси в мікрокоманді необхідно відвести 4 розряди.

В полі адреси буде розташовуватися адреса пам'яті – двійковий номер комірки, а в полях Y_1, Y_2, Y_3 та x – коди мікрооперацій та логічних умов. Остаточний формат мікрокоманди буде мати вигляд, який наведений на рисунку 11.3.

Y_1	Y_2	Y_3	x	$A1$
3	3	2	2	4

Рисунок 11.3. Формат мікропрограми

Кодування мікрооперацій та логічних умов будемо здійснювати довільно, наприклад так, як показано в таблиці 11.1.

Вибравши кодування, можна починати писати мікропрограму в машинних мікрокодах. Фактично формуємо вміст ПЗП мікропрограм (таблиця 11.2).

Аналізуючи ГСА мікропрограми (див. рисунок 11.1), бачимо, що в першому такті роботи автомата повинні бути видані мікрооперації y_2 та y_5. Враховуючи, що в початковому стані автомата Рг Лч А МК зручно встановити в 0, мікрокоманда, розташована за нульовою адресою, повинна сформувати мікрооперації y_2 та y_6. З таблиці 11.1 слідує, що $y_2 \in Y_1$ має код 010, а $y_6 \in Y_2$ (код 011). Мікрооперації, включені в множину в цій мікрокоманді відсутні. Тоді поле мікрооперацій мікрокоманди має містити код: 010 011 00.

Таблиця 11.1. Таблиці кодування мікрооперацій та логічних умов

Код	Y_1	Y_2	Y_3	Код	x
000	$\varnothing$	$\varnothing$	$\varnothing$	00	Константа 0
001	y_1	y_4	y_7	01	x_1
010	y_2	y_5	y_8	10	x_2
011	y_3	y_6	y_{13}	11	Константа 1
100	y_{10}	y_9			
101	y_{11}	y_{12}			
110		$y_к$			
111					

Після першої операторної вершини ГСА слідує умовна вершина, що містить логічну змінну x_1. Отже, в мікрокоманді повинна аналізуватися змінна

x_1. При $x_1 = 0$ наступною повинна виконуватися мікрокоманда (y_1, y_4, y_7) за адресою 1, інакше – мікрокоманда (y_1, y_5, y_8) за невідомою поки адресою. Заповнимо рядок таблиці для нульової комірки пам'яті таким кодом: 010 011 00 01 ????. До цього рядка ще повернемося для заповнення поля адреси переходу.

За адресою 1 повинна розташовуватися мікрокоманда, що формує мікрооперації (y_1, y_4, y_7) та передає керування наступній мікрокоманді (y_3, y_5). Заповняємо рядок таблиці для першої комірки пам'яті: 001 001 01 00 *xxxx*. В полі x цієї мікрокоманди код 00 вказує на тотожно хибну умову (константу 0) – до Рг Лч А МК буде додана 1, а вміст поля адреси переходу не використовується.

Продовжуючи аналогічним чином, заповняємо рядки таблиці 11.2, що відповідають адресам ПЗП 3, 4, 5, 6 (на рисунку 11.3 поруч з операторними вершинами позначені адреси відповідних мікрокоманд). У мікрокоманді за адресою 4 виконується умовний перехід по змінній x_2, причому адреса переходу при $x_2 = 1$ поки також невідома. За адресою 6 розташовується мікрокоманда, що відповідає кінцевій вершині ГСА, що завершує роботу мікропрограми мікрооперацією $y_к$. Таким чином, завершено мікропрограмування ділянки ГСА від початкової до кінцевої вершини, що відповідає нульовим значенням логічних змінних.

Тепер повернемося до логічної вершини, що розташована після першої операторної. Мікрокоманду, яка слідує за її одиничним виходом (y_1, y_5, y_8), можна розташувати за наступною вільною (7) адресою. Тому в поле адреси переходу комірки 0 тепер можна помітити код 0111. Сама мікрокоманда за адресою 7 формує три мікрооперації та переходить до мікрокоманди за адресою 8: 001 010 10 00 *xxxx*.

Мікрокоманда за адресою 8 не зв'язана з логічною вершиною, але вона повинна передавати керування вже існуючій (за адресою 3) мікрокоманді. Тому її поле $x = 11$ адресує тотожно істинну умову, а в полі переадресації вказана адреса 3.

Таблиця 11.2. Вміст ПЗП мікропрограм

Адреса	Y_1	Y_2	Y_3	x	$A1$
0	010	011	00	01	0111(7)
1	001	001	01	00	*xxxx*
2	011	010	00	00	*xxxx*
3	000	001	00	00	*xxxx*
4	101	101	11	10	1001 (9)
5	100	000	00	00	*xxxx*
6	000	110	00	00	*xxxx*
7	001	010	10	00	*xxxx*
8	010	100	00	11	0011 (3)
9	000	000	00	01	0111 (7)
10	000	000	00	11	0001 (1)

Нарешті, залишилася невизначеною адреса переходу в мікрокоманді за адресою 4. на даному етапі вже всім операторним вершинам ГСА (включаючи кінцеву) відповідають мікрокоманди в комірках ПЗП. Визначимо, на яку з них необхідно передати керування після перевірки умови x_2, якщо вона виявиться істиною. З ГСА видно, що тоді необхідно перевірити умову x_1 – випадок двох підряд розташованих умовних вершин. Передати керування на адресу 0, де перевіряється ця умова не можна, бо тоді виконаються мікрооперації y_1, y_4, y_7, а це мікропрограмою не передбачено. Очевидно, в мікропрограму необхідно включить додатково мікрокоманди (у нашому випадку – за адресою 9 та 10), які не формують ніяких мікрооперацій, а забезпечують тільки передачу керування. Перша здійснює умовний перехід по змінній x_1 на адресу 7, друга (яка буде виконуватися тільки при $x_1 = 0$) – безумовно на адресу 1. Тепер код мікропрограми повністю сформований.

Структурну схему розробленого керуючого автомата подано на рисунку 11.4.

ЛЕКЦІЯ 14

ОСНОВНІ ПОНЯТТЯ КОНТРОЛЮ КОМБІНАЦІЙНИХ СХЕМ

Розрізняють два методи контролю цифрових пристроїв: тестовий та функціональний.

Тестовий контроль здійснюється за допомогою спеціальних систем технічного діагностування, особливістю яких є можливість подання на об'єкт контролю спеціально організованих (тестових) впливів – двійкових наборів, які дають змогу виявляти несправності, які є в його схемі. Тестовий контроль переважно використовується тоді, коли об'єкт контролю не використовується за прямим призначенням. Але тестовий контроль може також застосовуватися для діагностики об'єкту контролю в процесі його функціонування, якщо в цьому процесі можна виділити такі моменти, коли на входи об'єкту контролю не надходять робочі вхідні сигнали та виходи можуть бути відключені від об'єктів керування. Основна задача тестового контролю – виявлення в схемі об'єкта контролю всіх несправностей із заданого класу з можливістю позначення місця розташування неправильно функціонуючих елементів.

Структурна схема тестового контролю наведена на рисунку 14.1. В такій структурі запам'ятовуючий пристрій (ЗП) зберігає тести та еталонні реакції, які надходять на входи об'єкту контролю та аналізатора за командами зі сторони схеми керування. Результати тестування використовуються для організації взаємозв'язку між об'єктом контролю та об'єктом керування.

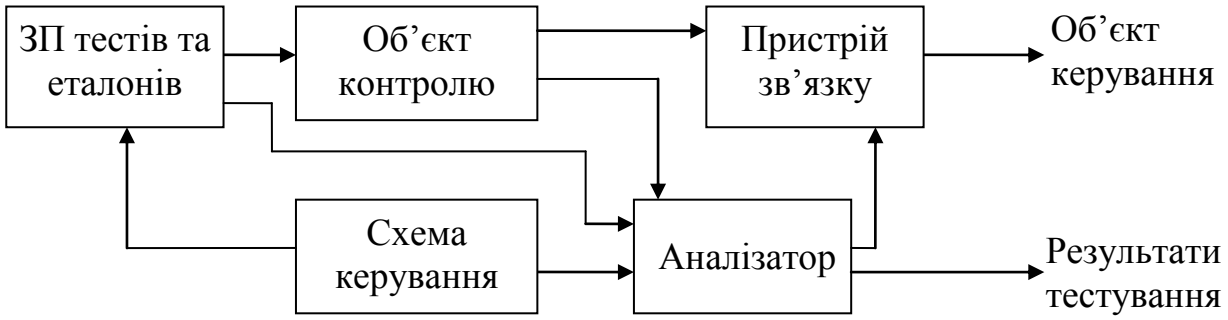


Рисунок 14.1. Структурна схема тестового контролю

Можливі два варіанти організації тестового контролю. Перший з них передбачає включення основного робочого режиму, після чого подається тест перевірки. Отриманий при виконанні робочого режиму результат запам'ятовується до завершення процесу проходження тесту. Якщо тест не фіксує наявність несправностей, то дається дозвіл на подальше використання вказаного результату. Згідно другого варіанту організації тестового контролю спочатку здійснюється подача тесту, а потім реалізується робочий режим. Вихідний результат роботи об'єкту контролю оцінюється за результатом тесту.

Основною задачею тестового контролю є проектування *тестів* - вхідних впливів, що забезпечують знаходження та локалізацію несправностей об'єктів контролю.

Для синтезу тестів необхідне представлення об'єкту з деякою степеню деталізації його структури та орієнтацію на певний математичний апарат. Окрім цього необхідна модель несправностей, яка достатньо адекватно описує физико-технологічні процеси, що відбуваються в ході експлуатації об'єкту контролю. Методи синтезу тестів, що базуються на детальній структурній інформації про об'єкт контролю називаються структурними.

Якість синтезованого тесту визначається його параметрами. Розрізняють довжину (кількість векторів тесту), повноту (відношення кількості знайдених тестом несправностей до сумарної кількості несправностей об'єкта контролю), складність реалізації тесту та його компактність. Оптимальний тест має мати мінімальну довжину, володіти максимальною повнотою, мінімальною складністю реалізації, а також бути максимально компактним.

За призначенням тести поділяють на перевіряючі та діагностуючі. *Перевіряючий тест* – це сукупність перевірок, що дозволяє виявити в системі будь-яку несправність із заданого списку (множини). Перевіряючий тест вирішує задачі перевірки справності системи (в цьому випадку у список несправностей включають всі можливі в системі несправності) та перевірки роботоздатності (до списку включають тільки ті несправності, що призводять до відмови системи). *Діагностуючий тест* – це сукупність перевірок, що дозволяє вказати місце несправності з точністю до класів еквівалентних несправностей. Під еквівалентними несправностями розуміють такі несправності, які не можна відрізнити одну від іншої при вибраному способі діагностування, що використовується для дослідження.

Важливою характеристикою процедур діагностування є повнота виявлення несправностей, що вказує долю несправностей, що гарантоварно виявляються, відносно всіх заданих або розглянутих несправностей об'єкта діагнозу. Будь-яка діагностична процедура (а також й тест діагнозу) обов'язково пов'язана із визначеним, строго фіксованим списком несправностей, виявлення яких забезпечується при її проведенні. Це фактично визначає обмеження, що накладаються на процес виявлення несправностей, та в кінцевому випадку визначає глибину діагностування.

За повнотою виявлення несправностей розрізняють поодинокі, кратні та повні тести. *Поодинокий тест* виявляє в пристрої всі поодинокі пошкодження елементів, що входять в нього. *Кратний тест* виявляє всі можливі сукупності з k поодиноких несправностей елементів, причому тест кратності k має фіксувати не тільки всі сукупності з k поодиноких несправностей, але й всі несправності меншої кратності, в тому числі всі поодинокі несправності. *Повний тест* виявляє несправності будь-якої кратності. Використання того чи іншого тесту визначається задачею діагнозу, що розв'язується. Так, при дослідженні пристрою, в якому виникла несправність в процесі функціонування, як правило, використовують поодинокі тести, оскільки ймовірність виникнення одночасно декількох несправностей не велика. У

порівнянні з поодинокими повні тести мають значно більшу довжину та тому вимагають для дослідження пристрою значно більше часу. Їх використовують при контролі пристроїв в процесі виготовлення, коли ймовірність одночасного існування декількох пошкоджень збільшується через дефекти комплектуючих виробів та помилок в монтажі та налаштуванні.

Залежно від довжини розрізняють тривіальний, мінімальний та мінімізований тести. *Тривіальний тест*, що містить всі можливі для даної системи перевірки, має максимальну довжину. Застосування тривіального тесту передбачає повне моделювання роботи пристрою. Найменшу кількість перевірок має *мінімальний тест*. Він забезпечує розв'язання заданої задачі діагнозу, при цьому для даного пристрою не існує іншого тесту з меншою кількістю перевірок.

Побудова мінімального тесту вимагає великих обчислень, тому на практиці часто будуються мінімізовані тести, що мають довжину, яка є близькою до довжини мінімальних тестів.

За допомогою тесту будується процедура діагностування, в основі якої лежить *алгоритм діагностування*, що являє собою послідовність елементарних перевірок, що складають тест, та правила аналізу результатів цих перевірок. Алгоритм діагностування реалізується засобами діагностування.

Функціональний контроль здійснюється тоді, коли об'єкт безпосередньо реалізує призначений йому алгоритм функціонування (об'єкт застосовується за призначенням). Можливе також використання функціонального контролю перед або після використання об'єкта за призначенням. Однак у цьому разі може бути потрібна імітація режиму функціонування об'єкта, оскільки кожен екземпляр об'єкта контролю має свою вбудовану апаратуру функціонального контролю. Функціональний контроль ще називають вбудованим контролем.

Укрупнену схему системи функціонального контролю можна подати у вигляді двох блоків (рисунок 14.2): об'єкту контролю та схеми контролю, де об'єднана вся додаткова апаратура, що може включати блок керування, блок моделі об'єкту діагнозу, блок розшифровки результатів, генератор сигналів тощо. Результатом контролю є сигнал помилки, який формується при виникненні дефекту в об'єкті контролю, а також, можливо, й в самій схемі контролю.

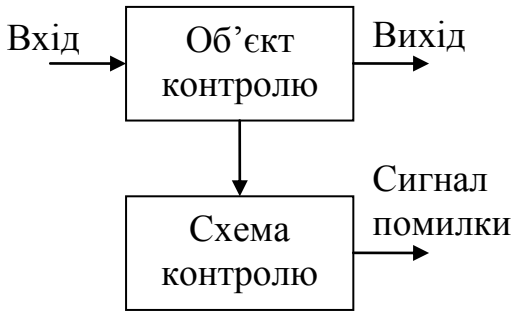


Рисунок 14.2. Блок-схема функціонального контролю

При організації функціонального контролю виникає питання справності самої схеми контролю. Це питання вирішується за рахунок наданню цифровому автомату властивості самоперевірки. Під властивістю самоперевірки розуміють здатність системи виявляти помилки як в об'єкті контролю, так й у схемі контролю, в процесі нормального функціонування без додаткового подання на входи пристрою спеціальних тестів перевірки або інших методів його дослідження. На рисунках 14.3 та 14.4 подані дві можливі структури самоперевіряючих цифрових автоматів. В першому випадку (рисунок 14.3) в якості схеми контролю використовується самоперевіряюча схема вбудованого контролю (ССВК), що має два контрольних виходи z_1 та z_2. На входи ССВК подаються вхідні та вихідні сигнали цифрового автомата. ССВК володіє наступною властивістю: для всіх можливих вхідних сигналів при справних цифровому автоматі та ССВК на виходах z_1 та z_2 формуються парафазні сигнали 01 або 10, а при виникненні несправностей в цифровому автоматі або ССВК – непарафазні сигнали 00 або 11. Для забезпечення функціонування структури на рисунку 14.3 початковий цифровий автомат не змінюється, до нього просто додається ССВК.

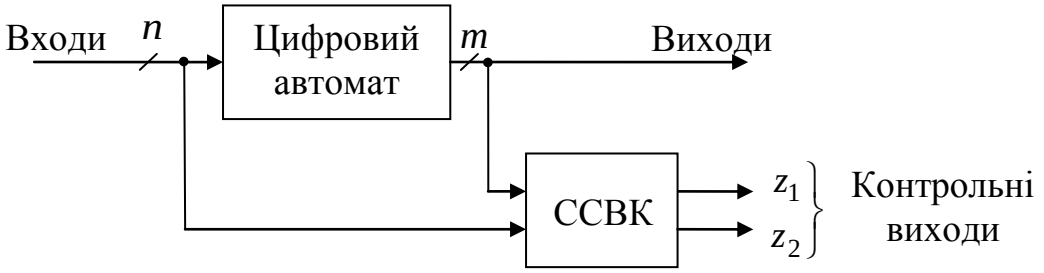


Рисунок 14.3. Перша структура самоперевіряючого цифрового автомата

Характерною особливістю другої структури самоперевіряючого цифрового автомата (рисунок 14.4) є те, що модифікується схема самого цифрового автомата, з метою фіксування несправності тільки за значеннями основних або спеціальних контрольних виходів. Як правило, на цих виходах формуються двійкові вектори, що належать деякому коду із виявленням помилок. Тоді ССВК синтезується у вигляді тестера, задача якого полягає у визначенні факту приналежності кодового вектора заданому коду.

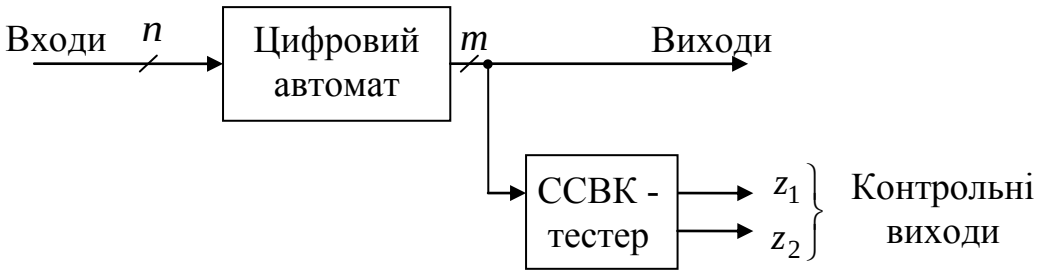


Рисунок 14.4. Друга структура самоперевіряючого цифрового автомата

Проведення тестового та функціонального контролю вимагає введення надлишковості або в схему об’єкта контролю (структурна надлишковість), або в інформацію, яка подається на вході об’єкта контролю (інформаційна надлишковість), або в проміжок часу, протягом якого здійснювався контроль (часова надлишковість). Тестовий контроль, як правило, пов’язаний із використанням інформаційно-часової надлишковості (для повірки правильності функціонування об’єкта контролю використовуються спеціальні тестові послідовності, що подаються на входи об’єкта протягом часу, який відводиться на контроль). Функціональний контроль пов’язаний із використанням перш за все структурної надлишковості.

Організація контролю цифрових автоматів пов’язана з необхідністю вивчення причин їх неправильної роботи. Основною причиною є несправності елементів схем цифрових автоматів. Розрізняють два види несправностей: відмови та збої.

Під *відмовою* розуміють безповоротне порушення характеристик окремих елементів схеми цифрового автомата, що призводить до повної втрати працездатності елемента. Якщо спотворюються характеристики кількох елементів, то йдеться про кратні відмови. При відмові для відновлення працездатності необхідно усувати несправності в схемі.

Під *збоєм* розуміють подію, яка полягає в тимчасовій зміні характеристик окремого елемента схеми цифрового автомата, що призводить до його неправильного функціонування протягом короткого інтервалу часу. Працездатність елемента відновлюється автоматично, без втручання ззовні. Після збою цифровий автомат може тривалий час працювати нормально. Збої супроводжуються викривленнями інформації при операціях передавання, зберігання та її обробки. Отже, якщо не усунути наслідки збою, то задача може виявитися невірно розв’язаною через викривлення даних або проміжних результатів. При збої відновлюється лише достовірність інформації.

У схемах цифрових автоматів причиною відмов може служити порушення граничних електричних норм застосування елементів, викликане або аварійною ситуацією, або неправильною експлуатацією, а також прихованими дефектами елементів, мікросхем та монтажу. Основними причинами збоїв є внутрішні та зовнішні завади, розкид параметрів елементів та вихід за норми допусків, що викликані довготривалим зберіганням, старінням тощо.

ЛЕКЦІЯ 15

МЕТОДИ ПОБУДОВИ ТЕСТІВ ДЛЯ КОМБІНАЦІЙНИХ СХЕМ

Метод таблиці функцій несправностей

Побудова тестів методом таблиць функцій несправностей при врахуванні лише поодиноких несправностей проводиться згідно етапів:

1. На комбінаційній схемі помічаються місця можливих несправностей.
2. Складається таблиця несправностей, що містить перелік несправностей та їх значення.
3. Будується таблиця функцій несправностей для поодиноких несправностей.
4. Будується скорочена таблиця функцій несправностей, шляхом викреслювання з таблиці функцій несправності стовпців, що відповідають еквівалентним несправностям та несправностям, що не можуть бути виявлені даним методом діагностування.
5. За скороченою таблицею функцій несправностей розраховується перевіряючий тест.
6. Будується таблиця покриттів, стовпці якої відповідають парам несправностей, що можна розрізнити.
7. Визначається діагностичний тест за таблицею покриттів.

Недолік методу таблиць функцій несправностей полягає у великих розмірах. Це пояснюється тим, що ТФН описує схему повністю та збільшення кількості елементів схеми пропорційно збільшує кількість стовпців таблиці. Тому для схем великих розмірів використовуються методи, в яких аналіз схеми проводиться по частинах, зокрема, так звані локальні алгоритми.

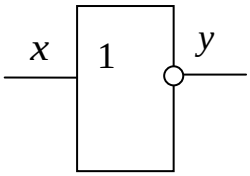
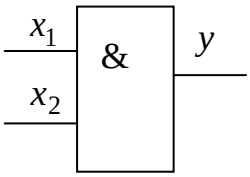
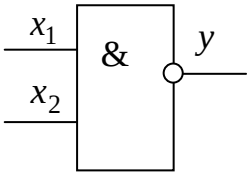
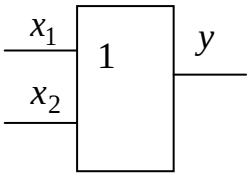
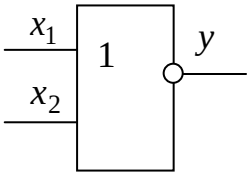
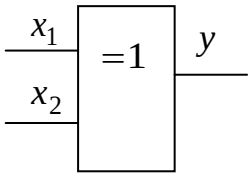
Метод суттєвих шляхів

При застосуванні локальних алгоритмів не використовують логічні функції, що реалізуються всією схемою або окремою підсхемою, а оперують лише інформацією, що характеризує елементи схеми. За допомогою локального алгоритму розв'язується наступна задача: чи є даний вхідний набір X перевіряючим для заданої константою несправності лінії i схеми (несправність n_i^{α}, $\alpha = 0$ або 1). Шляхом багаторазового застосування локального алгоритму для кожної несправності обраховується функція перевірки. За множиною функцій перевірки визначається перевіряючий або діагностуючий тести.

Згідно даного методу схема задається структурно, а саме списком елементів, вказанням типу елемента та зв'язків елементу з іншими елементами. Алгоритм опрацьовує кожен елемент окремо (локально) та всі елементи поступово, починаючи з елементів першого рангу або рівня, входи яких з'єднані тільки із входами схеми, та закінчуючи вихідними елементами. Входи та виходи схеми розглядаються як елементи схеми.

Для опису роботи логічних елементів складаються стислі таблиці істинності (таблиця 15.1). Символ x в таблицях має сенс довільного значення ($x = 0$ або 1). Кожен рядок таблиці називається *кубом*. Наприклад, в таблиці елемента І-НІ куб $0x1$ має зміст: якщо на одному вході елемента І є сигнал 0, то незалежно від значення на другому вході сигнал на виході дорівнює 1.

Таблиця 15.1. Стислі таблиці істинності логічних елементів

Графічне позначення логічного елемента	Назва логічного елемента	Стисла таблиця істинності
	НІ	x | y 0 | 1 1 | 0
	2І	x_1 | x_2 | y 1 | 1 | 1 x | 0 | 0 0 | x | 0
	2І-НІ	x_1 | x_2 | y 1 | 1 | 0 x | 0 | 1 0 | x | 1
	2АБО	x_1 | x_2 | y 0 | 0 | 0 x | 1 | 1 1 | x | 1
	2АБО-НІ	x_1 | x_2 | y 0 | 0 | 1 x | 1 | 0 1 | x | 0
	суматор за модулем 2	x_1 | x_2 | y 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1

Кожен логічний елемент характеризується також визначеними умовами передачі (трансляції) на вихід помилок сигналів на входах. Наприклад, у елемента АБО (рисунок 15.1) вхідна помилка транслюється на вихід при умові, що на другий вхід надходить сигнал 0. Помилки типу $1\to 0$ та $0\to 1$ позначають символами D та $\overline{D}$ відповідно. Умови трансляції помилок через елемент АБО задаються таблицею 15.2. Рядок таблиці називається D-кубом, кожен з яких задає одну з можливих умов трансляції помилки. Наприклад, запис кубу $D0D$ означає: якщо на одному вході існує помилка $1\to 0$, а на іншому сигнал 0, то на виході буде помилка D.

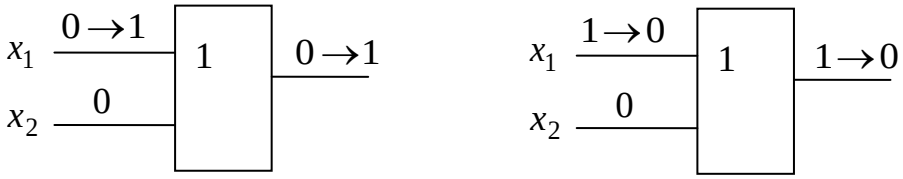


Рисунок 15.1. Трансляція вхідних помилок елемента АБО

Таблиця 15.2. Умови трансляції помилок для елемента АБО

x_1	x_2	y
D	0	D
0	D	D
$\overline{D}$	0	$\overline{D}$
0	$\overline{D}$	$\overline{D}$
D	D	D
$\overline{D}$	$\overline{D}$	$\overline{D}$

Таблиця 15.3. Умови відсутності трансляції помилок для елемента АБО

x_1	x_2	y
D	1	1
1	D	1
$\overline{D}$	1	1
1	$\overline{D}$	1
D	$\overline{D}$	1
$\overline{D}$	D	1

Умови відсутності трансляції помилок задаються тупиковими D-кубами. Для елемента АБО вони представлені в таблиці 15.3. При використанні значення символа x ($x\in\{0,1,D,\overline{D}\}$) цю таблицю можна зменшити. Об'єднання стислої таблиці істинності та таблиць, що задають умови трансляції помилок, утворюють *таблиці D-кубів логічних елементів* (таблиці 15.4-15.9), що містять всю необхідну інформацію про елементи для розв'язку діагностичних задач.

Локальний алгоритм, що дозволяє будувати перевіряючи набори, будується з використанням D-кубів. Алгоритм відповідає також наступній вимозі: при його реалізації на i-тому кроці достатньо використовувати результати обчислень на $(i-1)$-му кроці.

Для того щоб на вхідному наборі X виявлялася несправність n_i^{α}, $\alpha=0,1$, необхідно й достатньо виконання двох умов.

Умова перша полягає в тому, що дана несправність повинна проявитись на лінії i. Це значить, що якщо на лінії i існує несправність К.0 (К.1), то на наборі X у виправленій схемі на лінії i має бути сигнал 1 (сигнал 0). Ця умова забезпечується *початковою фіксацією* сигналу у схемі, яка полягає у

приписуванні вхідним лініям схеми значень сигналів у відповідності з вхідним набором X, а також у приписуванні лінії i символу D, якщо на лінії i є несправність К.0, та символу $\overline{D}$ при несправності К.1.

Таблиця 15.4. Таблиця D-кубів для елемента НІ

x	y
0	1
1	0
D	$\overline{D}$
$\overline{D}$	D

Таблиця 15.5. Таблиця D-кубів для елемента І

x_1	x_2	y
1	1	1
x	0	0
D	1	D
$\overline{D}$	1	$\overline{D}$
D	D	D
$\overline{D}$	$\overline{D}$	$\overline{D}$
D	$\overline{D}$	0

Таблиця 15.6. Таблиця D-кубів для елемента АБО

x_1	x_2	y
0	0	0
x	1	1
D	0	D
$\overline{D}$	0	$\overline{D}$
D	D	D
$\overline{D}$	$\overline{D}$	$\overline{D}$
D	$\overline{D}$	1

Таблиця 15.7. Таблиця D-кубів для елемента І-НІ

x_1	x_2	y
1	1	0
x	0	1
D	1	D
$\overline{D}$	1	$\overline{D}$
D	D	$\overline{D}$
$\overline{D}$	$\overline{D}$	D
D	$\overline{D}$	1

Таблиця 15.8. Таблиця D-кубів для елемента АБО-НІ

x_1	x_2	y
0	0	1
x	1	0
D	0	$\overline{D}$
$\overline{D}$	0	D
D	D	$\overline{D}$
$\overline{D}$	$\overline{D}$	D
D	$\overline{D}$	0

Таблиця 15.9. Таблиця D-кубів для суматора за модулем 2

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1
D	1	$\overline{D}$
D	0	D
$\overline{D}$	1	D
$\overline{D}$	0	$\overline{D}$
D	D	0
$\overline{D}$	$\overline{D}$	0
D	$\overline{D}$	1

Умова друга полягає в тому, що повинен існувати принаймні один шлях від лінії i до вихідної лінії, такий, що для всіх елементів цього шляху виконується умова трансляції помилки, що викликана несправністю n_i^{α}. Це значить, що сигнал на виході несправної схеми буде відрізнятися від сигналу справленої схеми та, відповідно, помилка буде знайдена. Такий шлях називається *суттєвим*, та говорять, що він є *активізованим* на вхідному наборі X.

Для перевірки другої умови використовується *операція однозначного поширення* над елементами j-того рангу. При цьому елементи схеми ранжуються. Отже, операція однозначного поширення полягає у приписуванні

символа з множини $\{0,1,D,\overline{D}\}$ на виході кожного елемента j-того рівня за відомими символами на входах цього елемента. Виконання операції починається з елементів першого рівня та здійснюється відповідно до таблиць D-кубів.

Вхідна фіксація сигналів називається *суперечливою* відносно набору X, якщо в результаті виконання операції однозначного поширення на лінії i з символом D (або $\overline{D}$) з'явиться сигнал 0 (або 1). Це значить, що перша умова не виконується та даний набір не є перевіряючим.

Операція однозначного поширення є результативною, якщо вхідна фіксація сигналів не є *суперечливою* відносно набору X та якщо вихідній лінії виявиться приписаний символ D або $\overline{D}$ у результаті застосування операції однозначного поширення. Це значить, що друга умова виконується та набір X є перевіряючим.

Недоліком метод суттєвих шляхів є необхідність перебору варіантів при пошуку перевіряючих вхідних наборів.

D-алгоритм

Метод побудови тестів для комбінаційних схем згідно *D*-алгоритму полягає у побудові перевіряючого набору за структурою схеми без аналізу її роботи на вхідних наборах. При цьому вводиться поняття D-кубу несправності елемента, який утворюється з тестових наборів: приписується виходу логічного елемента сигнал D, якщо на тестовому наборі на виході є зміна сигналу $1\to 0$, та приписується сигнал $\overline{D}$, якщо існує зміна сигналу $0\to 1$. D-куби несправностей наведені в таблицях 15.10-5.15.

Таблиця 15.10. Таблиця D-кубів несправностей елемента НІ

x	y
0	D
1	$\overline{D}$

Таблиця15.11. Таблиця D-кубів несправностей елемента І

x_1	x_2	y
1	1	D
1	0	$\overline{D}$
0	1	$\overline{D}$

Таблиця 15.12. Таблиця D-кубів несправностей елемента АБО

x_1	x_2	y
0	0	$\overline{D}$
1	0	D
0	1	D

Таблиця 15.13. Таблиця D-кубів несправностей елемента І-НІ

x_1	x_2	y
1	1	$\overline{D}$
1	0	D
0	1	D

Таблиця15.14. Таблиця D-кубів несправностей елемента АБО-НІ

x_1	x_2	y
0	0	D
1	0	$\overline{D}$
0	1	$\overline{D}$

Таблиця 15.15. Таблиця D-кубів несправностей суматора за модулем 2

x_1	x_2	y
0	0	$\overline{D}$
0	1	D
1	0	D
1	1	$\overline{D}$

У загальному, *D*-алгоритм складається з двох етапів: *D-просування* та *зворотнє довизначення*.

На першому етапі проводиться «просування» символу D на вихід схеми, тобто утворюється хоча б один суттєвий для несправності *D*-шлях. Для цього розглядаються всі можливі шляхи від місця несправності до виходу схеми. Далі записується розширений *D*-куб несправності. При цьому вважається, що всі координати, окрім вже визначених в основному кубі, дорівнюють x. Задача *D*-алгоритму полягає в тому, щоб відповідно до логіки схеми замінити невизначені координати x на символ з множини $\{0,1,D,\overline{D}\}$ таким чином, щоби вихідна лінія схеми мала символ D або $\overline{D}$.

Окрім поняття D-кубу несправності елемента вводиться поняття D-перетину (позначається символом $\cap$) над елементами з множини $\{0,1,x,D,\overline{D}\}$. Нехай $a,b \in \{0,1,x,D,\overline{D}\}$. Тоді операція D-перетину задається такими рівностями:

1) співпадіння сигналів $a \cap a = a$;

2) довизначення сигналу $a \cap x = a$;

3) $a \cap b = \varnothing$, якщо $a \neq b$, $a \neq x$ та $b \neq x$ (символ $\varnothing$ позначає порожній перетин).

Операція D-перетину над кубами t_i та t_j полягає у застосуванні операції D-перетину над однаковими координатами векторів t_i та t_j. Перетин $t_i \cap t_j$ є порожнім, якщо хоча б одна координата вектора $t_i \cap t_j$ рівна $\varnothing$.

В результаті виконання першого етапу *D*-алгоритму визначається значення деяких вхідних змінних. Значення інших вхідних змінних знаходяться при виконанні другого етапу *D*-алгоритму – зворотнього довизначення.

Процедура довизначення починається із вихідного елемента, а потім розглядаються поступово елементи з номерами у зворотньому порядку, виходам яких приписані 0 або 1. При цьому розв'язується наступна задача. За відомим сигналом на виході елемента (0 або 1) визначаються сигнали на входах елемента (якщо вони не є визначені). Для цього виконується операція D-перетину отриманих в результаті виконання першого етапу *D*-алгоритму D-кубів із стислими кубами таблиць істинності елемента. Якщо отримане непорожній D-перетин, то він визначає вхідний набір, що перевіряє дану нестравність.

Отже, *D*-алгоритм забезпечує знаходження хоча б одного перевіряючого набору, якщо такий набір існує. При розгляді всіх можливих ланцюгів непорожніх перетинів обчислюються всі перевіряючі набори та формується перевіряюча функція.

Приклад . Визначити вхідний набір, що визначає несправність К.0 на виході елемента 6 – n_6^0 комбінаційної схеми на рисунку 15.2.

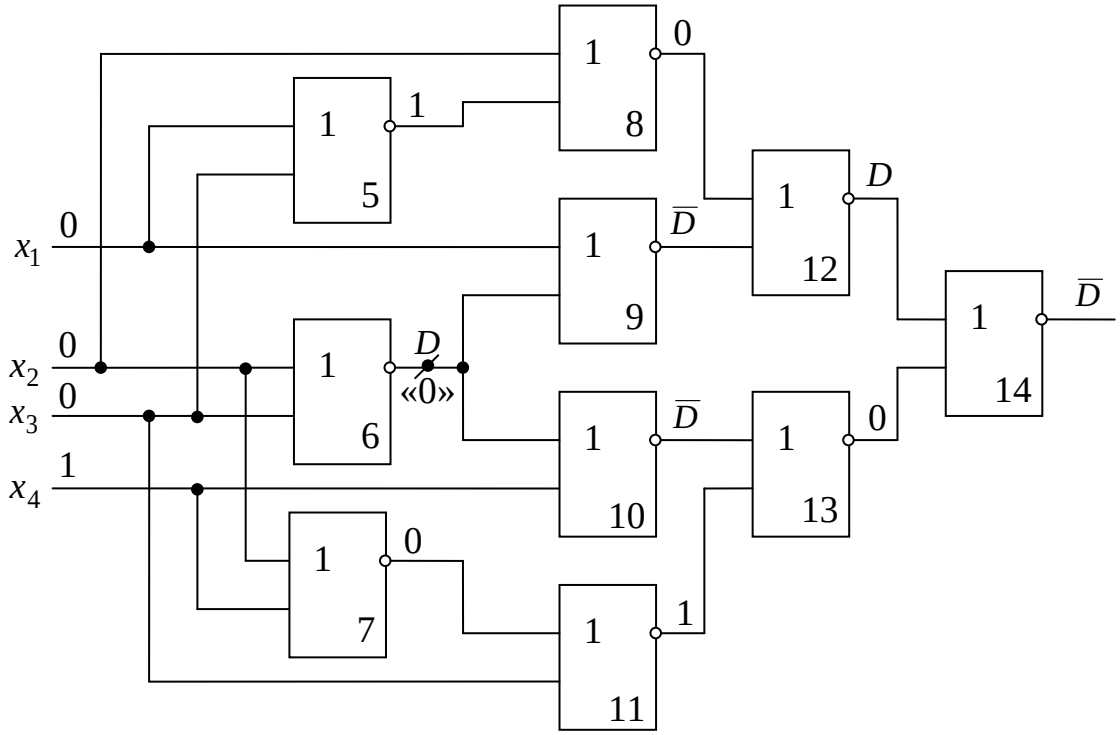


Рисунок 15.2.

Припишемо лінії 6 символ D, оскільки несправність n_6^0 виявляється шляхом зміни сигналу $1 \to 0$. D-куб несправності $t_6 = \dfrac{2\ \ 3\ \ 6}{0\ \ 0\ \ D}$ вказує, що для виявлення несправності n_6^0 на лінії 6 на лініях 2 та 3 мають бути присутні сигнали 0.

Згідно першого етапу D-алгоритму визначимо хоча б один суттєвий для несправності n_6^0 D-шлях. Будемо розглядати два шляхи: 6-9-12-14 та 6-10-13-14.

Запишемо розширений D-куб несправності. При цьому будемо вважати, що всі координати, окрім вже визначених в основному кубі, дорівнюють x:

$$t_6 = \frac{1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ \ 10\ \ 11\ \ 12\ \ 13\ \ 14}{x\ \ 0\ \ 0\ \ x\ \ x\ \ D\ \ x\ \ x\ \ x\ \ x\ \ x\ \ x\ \ x\ \ x}.$$

Далі відповідно до логіки схеми відшукаємо невизначені координати x.

Розглянемо шлях 6-9-12-14 та визначимо, чи він є суттєвим. Визначимо умови, коли символ $\overline{D}$ може з'явитися на лінії 9. Ці умови визначаються вказанням значень сигналів на лініях 1, 2, 3, 6, 9. Знаходимо непусті перетини D-кубу несправності t_6 та всіх D-кубів елемента 9. Останній є елементом АБО-НІ та має шість D-кубів:

№	1	6	9
1	$\overline{D}$	0	D
2	0	$\overline{D}$	D
3	D	0	$\overline{D}$
4	0	D	$\overline{D}$
5	$\overline{D}$	$\overline{D}$	D
6	D	D	$\overline{D}$

Вектор перетину кубу t_6 та D-кубу елемента 9 №1 має вигляд:

$$t^1_{6,9} = t_6 \cap t^1_9 = \frac{1\ 2\ 3\ 6\ 9}{x\ 0\ 0\ D\ x} \cap \frac{1\ 2\ 3\ 6\ 9}{\overline{D}\ x\ x\ 0\ D} = \frac{1\ 2\ 3\ 6\ 9}{\overline{D}\ 0\ 0\ \varnothing\ D}.$$

Перетин є порожнім, оскільки координата $6 = \varnothing$. Це значить, що існує протиріччя: на лінії 6 має бути сигнал D (за умовою куба t_6) й у той самий час – сигнал 0 (за умовою куба t^1_9).

Порожніми також є перетини кубу t_6 та D-кубів з номерами 2, 3, 5. Два куби перетину не є порожніми:

$$t^4_{6,9} = \frac{1\ 2\ 3\ 6\ 9}{0\ 0\ 0\ D\ \overline{D}} \text{ та } t^6_{6,9} = \frac{1\ 2\ 3\ 6\ 9}{D\ 0\ 0\ D\ \overline{D}}.$$

Куб $t^6_{6,9}$ має для вхідної лінії 1 значення D. Але вхідна лінія має бути визначена як 0 або 1. Тому виконується заміна у вхідній лінії символу D на символ 0 (оскільки $D = 1 \to 0$), а символа $\overline{D}$ на символ 1 (оскільки $\overline{D} = 0 \to 1$). Після такої заміни куб $t^6_{6,9}$ співпадає з кубом $t^4_{6,9}$. В результаті отримано один куб $t_{6,9} = \frac{1\ 2\ 3\ 6\ 9}{0\ 0\ 0\ D\ \overline{D}}$, який визначає наступні умови трансляції несправності n^0_6 на вихід елемента 9: на лініях 1, 2 та 3 має бути сигнал 0.

Далі здійснюється описана процедура послідовно для всіх елементів розглянутого шляху аж до останнього елемента схеми. В даному випадку розглядається елемент 12, при цьому знаходяться непорожні перетини отриманого раніше кубу $t_{6,9}$ та D-кубів елемента 12:

$$t^2_{6,9,12} = \frac{1\ 2\ 3\ 6\ 8\ 9\ 12}{0\ 0\ 0\ D\ 0\ \overline{D}\ D} \text{ та } t^5_{6,9,12} = \frac{1\ 2\ 3\ 6\ 8\ 9\ 12}{0\ 0\ 0\ D\ \overline{D}\ \overline{D}\ D}.$$

Далі знаходимо непорожні перетини даних кубів та D-кубів елемента 14. В результаті визначаються чотири непорожні куби, що визначають умови, за яких несправність транслюється на вихід схеми по суттєвому шляху 6-9-12-14:

		1	2	3	4	5	6	7	8	9	10	11	12	13	14
t_1	=	0	0	0			D		0	$\overline{D}$			D	0	$\overline{D}$
t_2	=	0	0	0			D		0	$\overline{D}$			D	D	$\overline{D}$
t_3	=	0	0	0			D		$\overline{D}$	$\overline{D}$			D	0	$\overline{D}$
t_4	=	0	0	0			D		$\overline{D}$	$\overline{D}$			D	D	D

Перейдемо до другого етапу – зворотнього довизначення. У розглянутому випадку необхідно визначити значення тільки одної вхідної змінної x_4.

Запишемо стислу таблицю істинності для елемента АБО-НІ (таблиця 15.16).

Таблиця 15.16. Стисла таблиця істинності для елемента АБО-НІ

Назва кубу	Входи		Вихід
	1	2	
g^1	0	0	1
g^2	x	1	0
g^3	1	x	0

Розглянемо отриманий вище куб t_1:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$t_1 =$	0	0	0			D		0	$\overline{D}$			D	0	$\overline{D}$.

В цьому кубі вихід та входи вихідного елемента схеми 14 визначені. Тому розглянемо елемент 13. На його виході має бути встановлений сигнал 0. Для знаходження значення сигналів на входах елемента 13 (лінії 10 та 11) визначаємо D-перетин куба t_1 зі стислими кубами елемента 13. Отримаємо два непорожні перетини:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$t_1 \cap g_{13}^2 =$	0	0	0			D		0	$\overline{D}$	x	1	D	0	$\overline{D}$,
$t_1 \cap g_{19}^2 =$	0	0	0			D		0	$\overline{D}$	1	x	D	0	$\overline{D}$.

Далі знаходимо непорожні перетини отриманих кубів зі стислими кубами елемента 11 й т.д. В результаті отримаємо куб:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$t_1 \cap g_{13}^2 \cap g_{11}^2 \cap g_8^2 \cap g_7^2 =$	0	0	0	1	1	D	0	0	$\overline{D}$	x	1	D	0	$\overline{D}$,

який визначає вхідний набір $x_1x_2x_3x_4 = 0001$, що перевіряє несправність n_6^0. При цьому активується D-шлях 6-9-12-14.

ЛЕКЦІЯ 16

МЕТОД ЕКВІВАЛЕНТНОЇ НОРМАЛЬНОЇ ФОРМИ. МЕТОД БУЛЕВИХ ПОХІДНИХ

Метод полягає на представленні булевої функції, що реалізує схему, у вигляді еквівалентної нормальної форми (ЕНФ). ЕНФ відрізняється тим, що її змінні співставленні не входам схеми, а всім можливим шляхам поширення сигналів. ЕНФ, як й звичайна нормальна форма, може бути обчислена методом підстановки, з тією лише різницею, що надлишкові терми не виключаються, оскільки вони характеризують конкретну реалізацію схеми. Для ЕНФ характерні такі особливості:

1. ЕНФ є логічною сумою логічних добутків, тобто нормальною формою булевої функції.

2. Аргументами ЕНФ є букви ЕНФ, під якими розуміють змінні або їх заперечення із індексом послідовності елементів відповідного шляху, що пов’язує цю змінну із виходом схеми.

3. Кількість букв ЕНФ у загальному випадку більше кількості вхідних змінних схеми, оскільки один й той самий вхід схеми може бути пов'язаний із виходом декількома шляхами.

4. ЕНФ може мати надлишкові кон’юнкції.

5. Якщо буква, що входить у ЕНФ, не має заперечення, то їй відповідає шлях із парною кількістю інверсій, у протилежному випадку – з непарною кількістю інверсій.

Приклад . Побудуємо ЕНФ для комбінаційної схеми з рисунку 16.1.

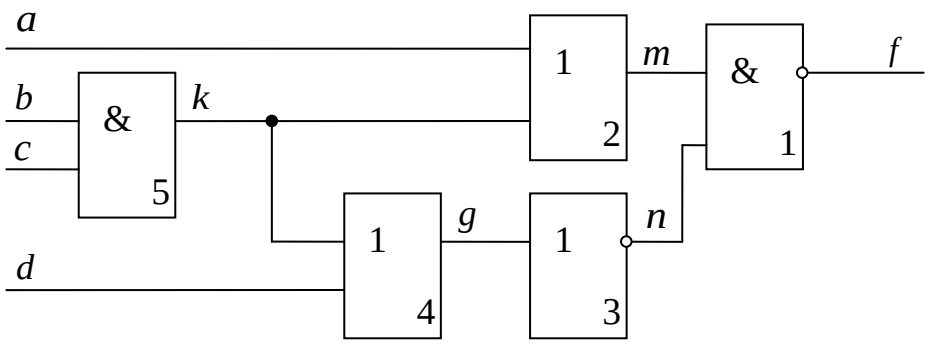


Рисунок 16.1.

ЕНФ будемо знаходити шляхом серій послідовних підстановок, яка починається з вихідного елементу схеми. Вихід елемента визначається через входи з вказанням індексу, що відповідає цьому елементу:

$$f = (\overline{m}\,\overline{n})_1.$$

Далі в отриманому виразі змінні, що відповідають виходам елементів 2 та 3, визначаються через входи аналогічним чином:

$$f = \left((\overline{a} + \overline{k})_2 (g)_3\right)_1.$$

Описаний процес продовжується допоки не будуть розглянуті всі елементи, тобто:

$$f = \left(\left(\bar{a} + \left(\bar{b}\,\bar{c}\right)_5\right)_2\left(\left(k + d\right)_4\right)_3\right)_1,$$

$$f = \left(\left(\bar{a} + \left(\bar{b}\,\bar{c}\right)_5\right)_2\left(\left(\left(bc\right)_5 + d\right)_4\right)_3\right)_1.$$

В отриманому виразі розкриваємо дужки при збереженні індексів елементів:

$$f = \bar{a}_{2,1}b_{5,4,3,1}c_{5,4,3,1} + \bar{b}_{5,2,1}\bar{c}_{5,2,1}b_{5,4,3,1}c_{5,4,3,1} + \bar{a}_{2,1}d_{4,3,1} + \bar{b}_{5,2,1}\bar{c}_{5,2,1}d_{4,3,1}.$$

Для спрощення запису ЕНФ послідовності елементів шляхів позначаємо цифрами:

2,1 – 1; 5,4,3,1 – 2; 5,2,1 – 3; 4,3,1 – 4.

Тоді ЕНФ можна записати у такому вигляді:

$$f = \bar{a}_1b_2c_2 + \bar{b}_3\bar{c}_3b_2c_2 + \bar{a}_1d_4 + \bar{b}_3\bar{c}_3d_4. \qquad (16.1)$$

Метод побудови тесту несправності за ЕНФ полягає в наступному.

Для кожної несправності комбінаційної схеми наводиться її проекція на ЕНФ у вигляді фіксації її букв в константи 0 або 1. На рисунку 16.2 наведено контрольну множину несправностей досліджуваної схеми. Для визначення проекції несправностей будується таблиця шляхів (таблиця 16.1). Рядки таблиці відповідають буквам ЕНФ, а стовпці – несправностям, що входять у контрольну множину.

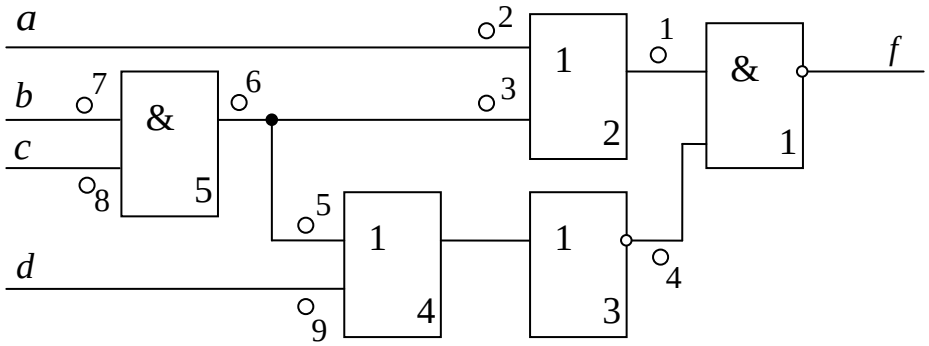


Рисунок 16.2. Контрольна множина несправностей

Таблиця 16.1. Таблиця шляхів

	Несправності								
	1	2	3	4	5	6	7	8	9
$\bar{a}_1$	0	0							
b_2				0	1	1	1		
$\bar{b}_3$	0		0			0	0		
c_2				0	1	1		1	
$\bar{c}_3$	0		0			0		0	
d_4				0					1

Таблиця шляхів заповнюється наступним чином:

1. на перетині j-го стовпця та i-го рядка в комірці таблиці ставиться 0 або 1, якщо несправність, що відповідає j-му стовпцю лежить на шляху, що відповідає i-му рядку;

2. в стовпці таблиці проставляється 1, якщо після несправності на даному шляху існує парна кількість інверсій, та ставиться 0 у протилежному випадку.

Проекція несправності за ЕНФ визначається за такими правилами:

1. несправність, що відповідає j-му стовпцю, фіксує в константи всі букви ЕНФ, у рядках яких в стовпці j проставлені 0 або 1;

2. якщо на перетині j-го стовпця та i-го рядка в комірці таблиці проставлена 1, то вид фіксації (0 або 1) i-ї букви ЕНФ відповідає виду несправності; якщо у вказаній комірці проставлений 0, то вид фіксації букви ЕНФ є протилежний до виду несправності.

Розглянемо, наприклад, несправність під номером 8 (рисунок 16.2). вона розташована на шляху, що з'єднує вхід c з виходом схеми через елементи 5, 2, 1 (шлях $\bar{c}_3$), а також на шляху, що з'єднує вхід c з виходом схеми через елементи 5, 4, 3, 1 (шлях c_2). На шляху c_2 після несправності 8 знаходиться парна кількість (дві) інверсій. Тому в таблиці 16.1 на перетині стовпця 8 та рядка c_2 проставлена 1. З цього слідує, що вид фіксації букв c_2 у виразі (16.1) співпадає з видом несправності 8 (константа 1). На шляху $\bar{c}_3$ після несправності 8 розташована непарна кількість (одна) інверсій. Тому в таблиці 16.1 на перетині стовпця 8 та рядка $\bar{c}_3$ проставлений 0. З цього слідує, що вид фіксації букв $\bar{c}_3$ у виразі (16.1) протилежний (константа 0) виду несправності 8.

В результаті для несправності 8 отримаємо наступну функцію несправності:

$$f_8 = \bar{a}_1 b_2 (c_2 = 1) + \bar{b}_3 (\bar{c}_3 = 0) b_2 (c_2 = 1) + \bar{a}_1 d_4 + \bar{b}_3 (\bar{c}_3 = 0) d_4 = \bar{a} b + \bar{a} d .$$

За функцією несправності визначається перевіряючий та діагностуючий тести.

Метод булевих похідних

Булевою похідною функції $f(x_1, \ldots, x_i, \ldots, x_n)$ відносно змінної x_i називається функція

$$\frac{df(x)}{dx_i} = f(x_1, \ldots, x_i, \ldots, x_n) \oplus f(x_1, \ldots, \bar{x}_i, \ldots, x_n).$$

Булеву похідну можна також обчислити згідно формули:

$$\frac{df(x)}{dx_i} = f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) \oplus f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n),$$

з якої видно, що булева похідна не залежить від змінної x_i, відносно якої вона обчислюється.

Як й будь-яка булева функція, похідна булевої функції приймає значення 0 або 1. При обчисленні похідних може виникнути один з трьох випадків:

1. якщо $\frac{df(x)}{dx_i} = g(x)$, то помилка в x_i буде викликати помилку на виході f, якщо $g(x) = 1$;

2. якщо $\frac{df(x)}{dx_i} = 0$, то помилка в x_i не буде викликати помилку на виході f, та, відповідно, функція $f(x)$ не залежить від змінної x_i;

3. якщо $\frac{df(x)}{dx_i} = 1$, то помилка в x_i завжди буде викликати помилку на виході f незалежно від значення інших вхідних змінних.

При обчисленні булевих похідних функцій, що мають невелику кількість змінних, використовують карти Карно. Для цього будуються дві карти для функцій $f(x_1,\ldots, x_i, \ldots, x_n)$ та $f(x_1,\ldots, \bar{x}_i, \ldots, x_n)$. Потім величини, що знаходяться у відповідних комірках карт, додаються за модулем 2. Результатом додавання є карта Карно, що відповідає похідній $\frac{df(x)}{dx_i}$.

При обчисленні булевих похідних складних функцій корисними є наступні їх властивості:

$$\frac{\overline{df(x)}}{dx_i} = \frac{df(x)}{dx_i};$$

$$\frac{d}{dx_i}\left(\frac{df(x)}{dx_j}\right) = \frac{d}{dx_j}\left(\frac{df(x)}{dx_i}\right);$$

$$\frac{d(f(x)g(x))}{dx_i} = f(x)\frac{g(x)}{dx_i} \oplus g(x)\frac{f(x)}{dx_i} \oplus \frac{df(x)}{dx_i}\cdot\frac{dg(x)}{dx_i};$$

$$\frac{d(f(x)+g(x))}{dx_i} = \overline{f(x)}\frac{g(x)}{dx_i} \oplus \overline{g(x)}\frac{f(x)}{dx_i} \oplus \frac{df(x)}{dx_i}\cdot\frac{dg(x)}{dx_i};$$

$$\frac{d(f(x)\oplus g(x))}{dx_i} = \frac{df(x)}{dx_i} \oplus \frac{dg(x)}{dx_i}.$$

ЛЕКЦІЯ 17

СТРУКТУРНІ СХЕМИ ФУНКЦІОНАЛЬНОГО КОНТРОЛЮ КОМБІНАЦІЙНИХ СХЕМ

Комбінаційна схема (КС) входить як складова частина в будь-який цифровий автомат, тому схема контролю КС завжди входить як складова частина у загальну схему контролю цифрового автомата. На рисунку 17.1 подана узагальнена структура функціонального контролю КС.

Основна схема, що підлягає контролю, $f(x)$ має n входів та m виходів. На виходах реалізуються функції $f_1(x)$, $f_2(x)$, …, $f_m(x)$, які утворюють робочі виходи схеми $y_1, y_2, ..., y_m$. При організації контролю основний блок $f(x)$ не підлягає змінам, а в структуру включається додатковий блок $g(x)$ (або блок контрольної логіки), на виходах якого реалізуються додаткові контрольні функції $g_1(x)$, $g_2(x)$, …, $g_k(x)$, $k \le m$. Третій блок структури – компаратор здійснює порівняння значень сигналів на виходах основного блоку $f(x)$ зі значеннями сигналів на виходах додаткового блоку $g(x)$. При виникненні несправностей, коли виникає невідповідність між значеннями вказаних сигналів, на виході z компаратора, який є контрольним виходом всієї структури, формується сигнал помилки.

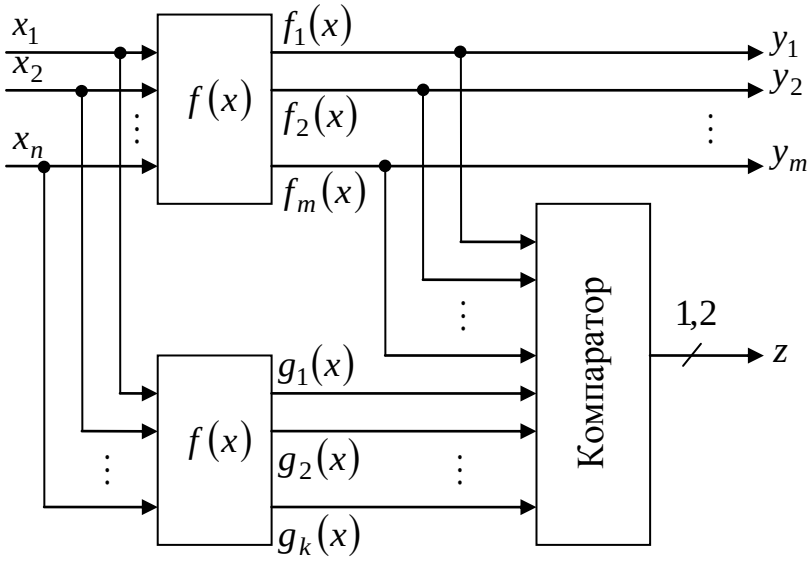


Рисунок 17.1. Структура функціонального контролю КС

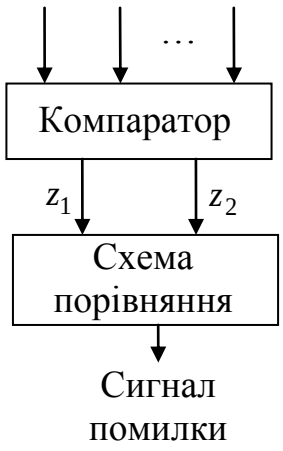


Рисунок 17.2. Схема контролю компаратора

Компаратор може мати один або два виходи. У самоперевіряючих цифрових автоматах, що реалізуються відповідно до рисунку 17.2, компаратор має два виходи z_1 та z_2. У цьому випадку в структуру включають четвертий блок – схему порівняння, який контролює наявність пара фазного сигналу на виходах компаратора (рисунок 17.2). В описаній структурі на виході схеми порівняння виявляються несправності, що виникають у всіх чотирьох блоках – $f(x)$, $g(x)$, компараторі та схемі порівняння. Оскільки схема порівняння

встановлюється одна для схеми будь-якої складності, в неї для забезпечення високої надійності може буту внесена суттєва надлишковість.

Можливі два підходи до побудови структури функціонального контролю. Перший підхід можна проілюструвати згідно рисунку 17.3. Він полягає в тому, що додатковий блок $g(x)$ обчислює такі функції $g_1(x)$, $g_2(x)$, …, $g_k(x)$, що на будь-якому вхідному робочому наборі на виходах блоків $f(x)$ та $g(x)$ формується вектор $\{f_1 f_2 ... f_m\ g_1 g_2 ... g_k\}$, який є словом деякого коду із виявленням помилок. Тоді можуть бути виявлені ті несправності всередині блоків $f(x)$ та $g(x)$, які викликають появу на виходах $f_1, f_2, ..., f_m, g_1, g_2, ..., g_k$ вектора, що не належить вибраному коду. У загальному випадку блок $g(x)$ обчислює контрольні розряди коду із виявленням помилок. Тому структуру на рисунку 17.3 називають схемою контролю методом обчислення контрольних розрядів. При цьому компаратор реалізується у вигляді тестеру, призначення якого полягає у тому, щоб зафіксувати факт приналежності кодового вектора $\{f_1 f_2 ... f_m\ g_1 g_2 ... g_k\}$ заданому коду. У протилежному випадку на виході тестера, що є контрольним виходом всієї схеми, формується сигнал помилки.

Другий підхід до побудови структури функціонального контролю КС поданий на рисунку 17.4. У цьому випадку додатковий блок $g(x)$ обчислює функції логічного доповнення $g_1(x)$, $g_2(x)$, …, $g_m(x)$, за допомогою яких функції основного блоку $f_1(x)$, $f_2(x)$, …, $f_m(x)$ перетворюються (доповнюються) у функції $h_1(x)$, $h_2(x)$, …, $h_m(x)$. Перетворення здійснюється таким чином, що на будь-якому вхідному наборі формується вектор $\{h_1 h_2 ... h_m\}$, який є словом деякого коду із виявленням помилок. Для перетворення використовуються суматори за модулем 2 (позначені на схемі як $\oplus$).

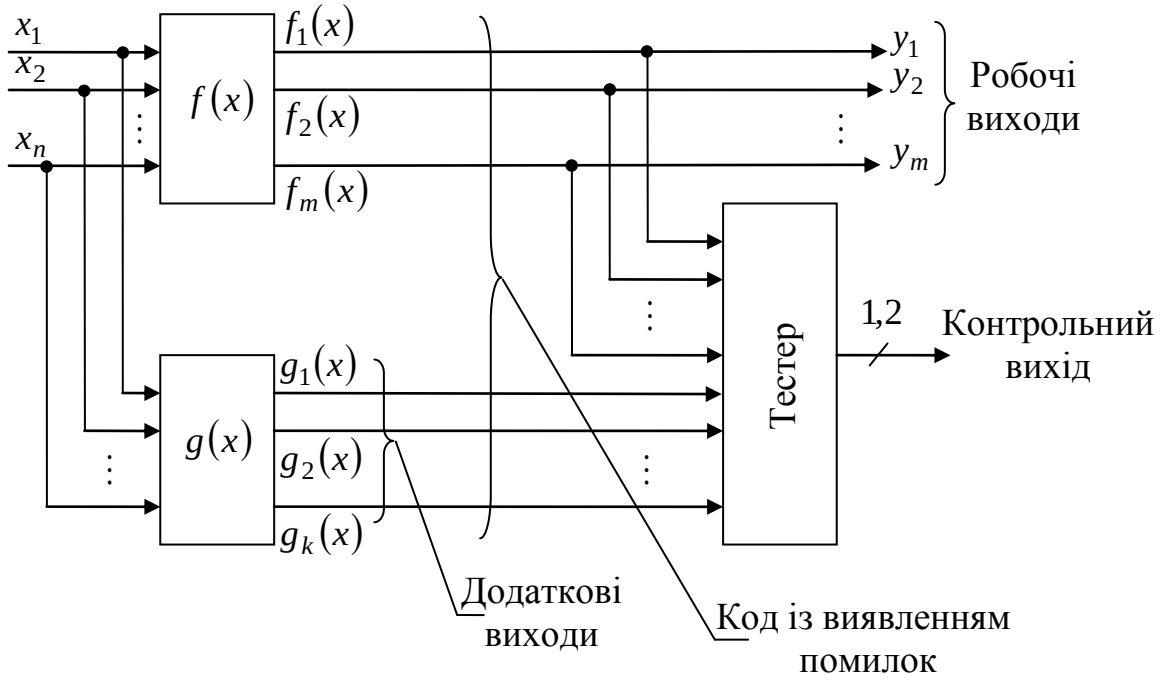


Рисунок 17.3. Схема контролю методом обчислення контрольних розрядів

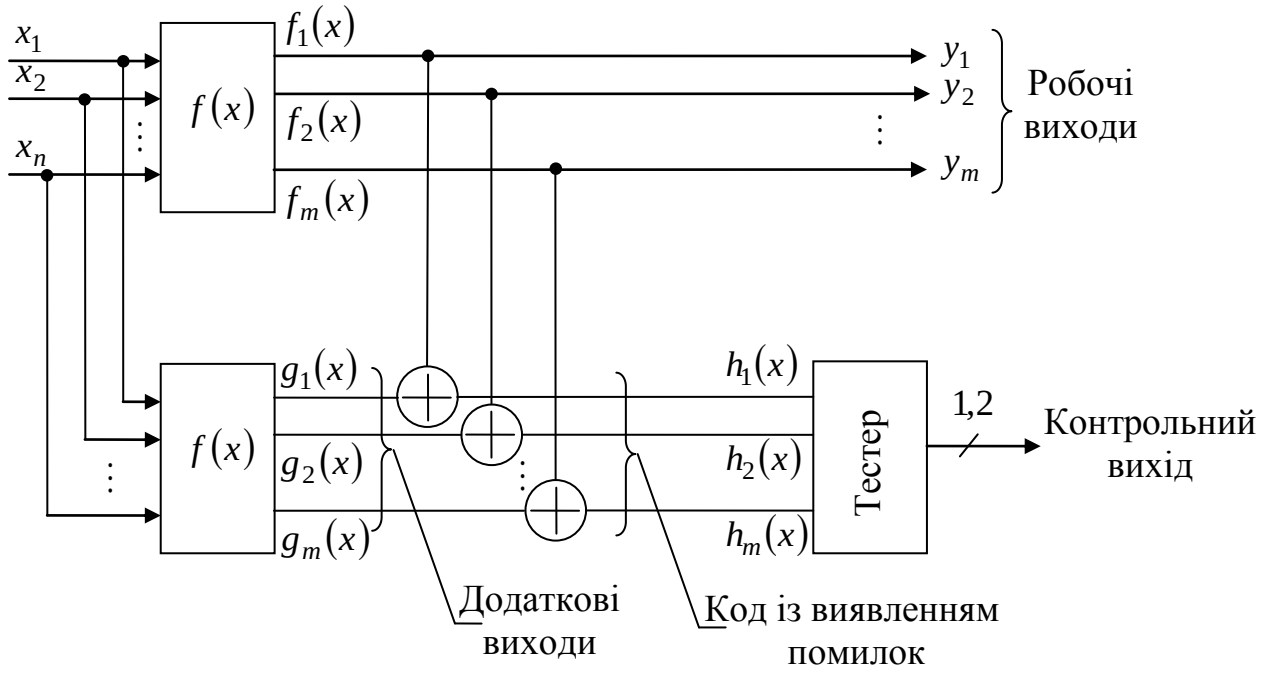


Рисунок 17.4. Схема контролю методом логічного доповнення

Тому

$$h_1(x) = f_1(x) \oplus g_1(x),$$
$$h_2(x) = f_2(x) \oplus g_2(x),$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$h_m(x) = f_m(x) \oplus g_m(x).$$

Компаратор містить в собі суматори за модулем 2 та тестер, що контролює вектори $\{h_1 h_2 ... h_m\}$. Така структура називається схемою контролю методом логічного доповнення.

Важливим елементом розглянутих структур функціонального діагностування КС є тестер. Задача тестера полягає в тому, щоб відрізняти кодові вектори, що належать розглянутому коду, від всіх інших можливих векторів. Переважно тестер має один вихід та описується функцією, що визначається через диз'юнкцію всіх векторів коду.

Отже, при побудові структур функціонального діагностування КС вирішуються наступні основні проблеми:

1. Синтез додаткового блоку $g(x)$ із найменшою складністю.

2. Забезпечення виявлення максимальної кількості несправностей в основному блоці $f(x)$, а також за можливістю й в блоці $g(x)$ та компараторі. При цьому вважається, що несправності в блоці $g(x)$ та компараторі не чинять безпосереднього впливу на робочі виходи $y_1, y_2, ..., y_m$ (рисунок 17.1) всього пристрою, а відповідно, й на роботу об'єктів керування. Тому забезпечення виявлення помилок на основному блоці $f(x)$ розглядається як основна задача.

3. Забезпечення повного контролю компаратора. Для розв'язання цієї задачі необхідно надходження на вхід компаратора перевіряючого тесту, що формується на виходах блоків $f(x)$ та $g(x)$.

4. Забезпечення необхідної швидкодії схеми контролю. Згідно рисунку 17.1, компаратор може виробляти сигнал помилки тільки після того, як на його входи надійдуть сигнали з виходів основного блоку $f(x)$, що також безпосередньо подаються на робочі виходи $y_1, y_2, ..., y_m$. Тому час формування сигналу помилки повинен бути менше часу сприйняття об'єктами керування сигналів з робочих виходів. При цьому вирішуючи значення має швидкодія компаратора.

Конкретні схеми функціонального контролю КС визначаються видом застосованого коду із виявленням помилок.

Метод дублювання

Метод дублювання при побудові цифрових автоматів часто використовується на практиці. У цьому методі в якості додаткового блоку $g(x)$ в структурі функціонального контролю встановлюється блок $f'(x)$, що цілком ідентичний основному блокові $f(x)$, тобто $f_1'(x) = f_1(x)$, $f_2'(x) = f_2(x)$, …, $f_m'(x) = f_m(x)$. Тоді при нормальній роботі на виходах основного та додаткового блоків формуються вектори кодів з повторенням. Компаратор складається із суматорів за модулем 2, які порівнюють сигнали на однакових виходах f_i обох блоків. Якщо для всіх $i \in \{1, 2, ..., m\}$ функції $f_i'(x) = f_i(x)$, то на виходах всіх суматорів за модулем 2 формуються сигнали 0, а відповідно, $z = 0$, що свідчить про відсутність несправностей. Якщо хоча б для одного i $f_i'(x) \neq f_i(x)$, то $z = 1$, та фіксується помилка.

Метод паритету

В основі методу паритету лежить використання коду з контролем на парність (непарність) або код паритету. В таблиці 17.1. проілюстрований принцип побудови коду паритету для випадку, коли кількість інформаційних розрядів дорівнює 3 – x_1, x_2, x_3. Код паритету містить всього один контрольний розряд x_4. Якщо в інформаційному векторі кількість одиниць парна, то $x_4 = 0$, якщо кількість одиниць непарна, то $x_4 = 1$. Таким чином, будь-який кодовий вектор містить парну кількість одиниць. В коді паритету виявляються всі помилки, які призводять до порушення парності числа одиниць, а саме всі помилки непарної кратності та, відповідно, всі поодинокі помилки.

Таблиця 17.1. Побудова коду паритету

$x_1x_2x_3$	x_4
0 0 0	0
0 0 1	1
0 1 0	1
0 1 1	0
1 0 0	1
1 0 1	0
1 1 0	0
1 1 1	1

ЛЕКЦІЯ 18

КОНТРОЛЬ ЗА КОДОМ ЗА ПОСТІЙНОЮ ВАГОЮ. КОНТРОЛЬ ЗА КОДОМ ІЗ ДОДАВАННЯМ. МЕТОД ЛОГІЧНОГО ДОПОВНЕННЯ

Код за постійною вагою (код «w з p», pCw-код) утворює двійкові p-розрядні вектори, в яких w розрядів мають значення одиниці. Кількість слів коду дорівнює C_p^w. У коді виявляються всі поодинокі та кратні однонаправлені помилки одного типу (або $0 \to 1$, або $1 \to 0$), оскільки вони змінюють кількість одиниць у кодовому векторі.

Схема контролю за кодом з постійною вагою відповідає схемі, що подана на рисунку 17.3. У цій схемі додатковий блок $g(x)$ обчислює додаткові функції $g_1(x)$, $g_2(x)$, …, $g_k(x)$ з таким розрахунком, щоб при поданні на вхід схеми будь-якого робочого вектора на виходах блоків $f(x)$ та $g(x)$ формувалися слова pCw-коду. При цьому $p = m + k$, а вага w залежить від максимальної кількості одиниць, яка може міститися у векторі $\{f_1 f_2 ... f_m\}$.

Розглянемо метод отримання коду. В таблиці 18.1 наведена таблиця істинності для функцій $f_1(x)$, $f_2(x)$, $f_3(x)$, з якої видно, що у справному стані на виходах схеми формуються три розрядні вектори з вагами $w = 0, 1, 2$. Кількість контрольних розрядів (додаткових виходів блоку $g(x)$) дорівнює:

$$k = w_{\max} - w_{\min},$$

де $w_{\max}$ та $w_{\min}$ – максимальна та мінімальна кількість одиниць у векторах, що формується на виходах справної схеми $f(x)$.

Таблиця 18.1. Таблиця істинності

$x_1 x_2 x_3$	f_1	f_2	f_3	g_1	g_2
0 0 0	1	0	0	1	0
0 0 1	0	1	0	1	0
0 1 0	0	0	1	1	0
0 1 1	0	0	0	1	1
1 0 0	1	1	0	0	0
1 0 1	0	1	1	0	0
1 1 0	1	0	0	1	0
1 1 1	1	0	0	0	1

Для контролю вибираємо код з вагою $w_{\max}$. У даному випадку $k = 2$, а тому може бути використаний $2C5$-код. У таблиці 18.1 у стовпцях g_1 та g_2 для кожного набору вхідних змінних здійснено довизначення вихідних векторів до векторів $2C5$-коду.

Контроль за кодом із додаванням

Код із додаванням (код Бергера, pSm-код ($p = m + k$)) відноситься до класу роздільних кодів із виявленням помилок. У слові коду виділяють інформаційну (m розрядів) та контрольну (p розрядів) частину. Кількість слів коду дорівнює 2^m. Множина інформаційних слів коду утворює всі можливі m-розрядні двійкові вектори. Контрольна частина коду визначається наступним чином. Підраховується кількість одиничних розрядів в інформаційному векторі кодового слова. Це число представляється у двійковому вигляді (допоміжне слово). Контрольне слово утворюється із допоміжного шляхом заміни у ньому одиничних розрядів на нульові та навпаки. Кількість контрольних розрядів розраховується за формулою:

$$k =]\log_2(m+1)[.$$

В інформаційному векторі виявляються однонаправлені помилки будь-якої кратності, оскільки при цьому змінюється кількість одиниць у кодовому векторі. Не виявляються кратні різнонаправлені помилки, за яких кількість одиниць у кодовому векторі зберігається. У контрольній частині кодового слова виявляються будь-які поодинокі помилки, оскільки при цьому завжди змінюється число, що записане у контрольному векторі.

Схема контролю за кодом із додаванням відповідає схемі, що подана на рисунку 17.3. У цій схемі додатковий блок $g(x)$ обчислює додаткові функції $g_1(x)$, $g_2(x)$, …, $g_k(x)$ з таким розрахунком, щоб при поданні на вхід схеми будь-якого робочого вектора на виходах блоків $f(x)$ та $g(x)$ формувалися слова pSm-коду ($p = m + k$). При цьому на виходах основного блоку $f(x)$ реалізуються інформаційні розряди, а на виходах додаткового блоку $g(x)$ – контрольні розряди коду. В якості компаратора використовується самоперевіряючих тестер pSm-СПТ.

Схема контролю за кодом із додаванням виявляє ті самі помилки в основному блоці $f(x)$, що й схема контролю за кодом з постійною вагою. Але її перевагою над схемою контролю за кодом з постійною вагою є те, що у додатковому блоці $g(x)$ виявляються всі можливі помилки.

Метод логічного доповнення

Недоліком схеми контролю методом обчислення контрольних розрядів (рисунок 10.6) є те, що ідея контролю, яка полягає в її основі, повністю копіює ідею виявлення помилок при передаванні інформації по лінії зв’язку. За цією причиною значення сигналів $g_1(x)$, $g_2(x)$, …, $g_k(x)$ однозначно визначається значеннями сигналів $f_1(x)$, $f_2(x)$, …, $f_m(x)$, та схема блоку $g(x)$ жорстко визначена та є переважно єдиною. Окрім того, задача отримання тесту перевірки на входах тестера при жорсткій залежності між сигналами $f(x)$ та

$g(x)$ важко вирішується, а у деяких випадках у рамках методу обчислення контрольних розрядів не має розв'язку.

Вказані недоліки в значній мірі усуваються у схемах контролю методом логічного доповнення (рисунок 17.4). У цьому випадку кількість розрядів коду із виявленням помилок, що контролюється тестером, дорівнює кількості виходів основного блоку $f(x)$. Тому у структурі логічного доповнення контролюється більш простіший код, що має m розрядів, ніж у структурі з обчисленням контрольних розрядів, де є $m+k$ розрядів. Це спрощує побудову самоперевіряючого тестеру, що буде мати менше входів та меншу складність.

Важливою характеристикою структури логічного доповнення є те, що блок $g(x)$ може мати багато варіантів побудови. Кожна функція $g_i(x)$ однозначно не визначається за значеннями функцій $f_1(x)$, $f_2(x)$, …, $f_m(x)$, оскільки вектор $\{f_1 f_2 ... f_m\}$ може бути перетворений у будь-який кодовий вектор $\{h_1 h_2 ... h_m\}$. При одному й тому самому значенні функцій $f_1(x)$, $f_2(x)$, …, $f_m(x)$ функція $g_i(x)$ може приймати різні значення. Це дозволяє при побудові структури (рисунок 10.7) здійснювати вибір між різними варіантами блоку $g(x)$ із найменшою складністю. Окрім того, шляхом підбору функцій $g_1(x)$, $g_2(x)$, …, $g_k(x)$ є можливість забезпечити надходження на входи тестера та суматорів за модулем 2 всіх наборів, що складають тест, який перевіряється. Тому метод логічного доповнення дозволяє будувати повністю самоперевіряючі структури контролю КС й у тих випадках, коли це не можливо методом обчислення контрольних розрядів.

ЛІТЕРАТУРА

1. Баранов, С.И. Синтез микропрограммных автоматов (граф-схемы и автоматы). Издание второе, переработанное и дополненное [Текст] / С.И. Баранов. – Ленинград : Энергия, 1979. – 232 с.
2. Бондаренко, М.Ф. Комп'ютерна дискретна математика: підручник [Текст] / М.Ф. Бондаренко, Н.В. Білоус, А.Г. Руткас. – Харків: Компанія СМІТ, 2004. – 480с.
3. Брауэр, В. Введение в теорию конечных автоматов [Текст] / В. Брауэр; перевод с английского под редакцией Ю.И. Журавлева. – М.: Радио и связь, 1987. – 392с.
4. Воробйова, О.М. Основи схемотехніки: у двох частинах: навч. посібник [Текст] / О.М. Воробйова, В.Д. Іванченко. – Одеса: ОНАЗ ім. О.С. Попова, 2004. – Ч. 2. – 172с.
5. Глушков, В.М. Синтез цифровых автоматов [Текст] / В.М. Глушков. – М.: Государственное издательство физико-математической литературы, 1962.
6. Голдсуорт, Б. Проектирование цифровых логических устройств [Текст] / М.: Машиностроение, 1985. – 288 с.
7. Дмитриев, В.И. Прикладная теория информации М.: 1989.
8. Жмакин А.П. Архитектура ЭВМ. СПб.: БХВ-Петербург, 2006. – 320 с.
9. Жураковський, Ю.П., Полторак, В.П. Теорія інформації та кодування: Підручник. – К.: Вища шк., 2001. – 255 с.
10. Емеличев, В.А. Лекции по теории графов [Текст] / В.А. Емеличев, О.И. Мельников. – М.: Наука, 1990.
11. Закревский, А.Д. Алгоритмы синтеза дискретных автоматов [Текст] / А.Д. Закревский. – М.: Наука, 1971. – 512 с.
12. Захаров, Н.Г. Синтез цифровых автоматов: учебное пособие [Текст] / Н.Г. Захаров, В.Н. Рогов. – Ульяновск: УлГТУ, 2003.
13. Каган Б.М. Электронные вычислительные машины и системы: Учеб. пособие для вузов. – 3-е изд., перераб. и доп. – М.: энергоатомиздат, 1991. – 592 с.
14. Карпов, Е.А. Теория автоматов [Текст] / Е.А. Карпов. – СПб.: Питер, 2003. – 208 с.
15. Кочубуй О.О., Сопільник О.В. Прикладна теорія цифрових автоматів. Логічні основи. Видавництво Дніпропетровського університету, 2009. - 264 с.
16. Кузьмин, И.В., Кедрус, В.А. Основы теории информации и кодирования – К.: Вища шк. Головное изд-во, 1986. – 238 с.
17. Липкин, И.А. Статистическая радиотехника. Теория информации и кодирования. – М.: «Вузовская книга», 2002. – 216 с.
18. Лупал, А.М. Теория автоматов: учеб. пособие [Текст] / А.М. Лупал. – СПб.: СПбГУАП, 2000. – 119 с.
19. Майоров С.А., Новиков Г.И. Структура электронных вычислительных машин. Л.: Машиностроение, Ленингр.отд-ие, 1979.

20. Микросхемы и их применение: Справочное пособие / В.А. Батушев, В.Н. Вениаминов, В.Г. Ковалев, О.Н. Лебедев. А.И. мирошниченко. – 2-е изд., перер. и доп. – М.: Радио и связь, 1983. – 272 с.

21. Наумкина Л.Г. Цифровая схемотехника. Конспект лекций по дисциплине «Схемотехника». – М.: Издательство «Горная книга», Издательство Московского государственного горного университета, 2008. – 308 с.

22. Нікольський, Ю.В. Дискретна математика: підручник [Текст] / Ю.В. Нікольський, В.В. Пасічник, Ю.М. Щербина. – Львів: Магнолія 2006, 2007. – 608с.

23. Основы технической диагностики. В 2-х книгах. Кн.1. модели объектов, методы и алгоритмы диагностики. Под ред. П.П. Пархоменко. М., «Энергия», 1976. – 464 с.

24. Питерсон, У., Уэлдон, Э. Коды, исправляющие ошибки. – М.: Мир, 1976. -

25. Поспелов, Д.А. Логические методы анализа и синтеза схем [Текст] / Д.А. Поспелов. – М.: Энергия, 1974.

26. Пухальский Г.И., Новосельцева Т.Я. Проектирование дискретных устройств на интегральных микросхемах: Справочник. – М.: Радио и связь, 1990. – 304 с.

27. Пухальский Г.И., Новосельцева Т.Я. Цифровые устройства: Учебное пособие для вузов. – СПб.: Политехника, 1996. – 885 с.

28. Савельев, А.Я. Основы информатики: учеб. для вузов [Текст] / А.Я. Савельев. – М.: Изд-во МГТУ им. Н. Э. Баумана, 2001. – 328 с.

29. Савельев, А.Я. Прикладная теория цифровых автоматов: учебник для вузов [Текст] / А.Я. Савельев. – М.: Высшая школа, 1987. – 272с.

30. Самофалов, К.Г. Прикладная теория цифровых автоматов [Текст] / К.Г. Самофалов. – Киев: Высшая школа, 1987.

31. Сапожников В.В., Сапожников Вл.В. Основы технической диагностики. М.: Маршрут, 2004. – 318 с.

32. Угрюмов, Е.П. цифровая схемотехника. – СПб.: БХВ-Петербург, 2004, - 528 с.

33. Уилкинсон, Б. Основы проектирования цифровых схем. – М.: Издательский дом «Вильямс», 2004. – 320 с.

34. Уэйкерли, Дж.Ф. Проектирование цифровых устройств. 2002. Т.1.

35. Фридман, А. Теория и проектирование переключательных схем [Текст] / А. Фридман, П. Менон. – М.: Мир, 1978.

36. Хэмминг, Р.В. Теория кодирования и теория информации. – М.: Радио и связь, 1983. – 176 с.

37. Хопкрофт, Д. Введение в теорию автоматов, языков и вычислений, 2-е изд.: пер. с англ. [Текст] / Д. Хопкрофт, Р. Мотванн, Д. Ульман. – М.: Вильямс, 2002. – 528 с.

38. Шульгин, В.И. Основы теории передачи информации. Ч. 2. помехоустойчивое кодирование. Учеб.пособие. – Харьков: Нац. Аэрокосм. Ун-т «Харьк. авиац. ин-т», 2003. – 87 с.

39. Якубовский, С.В. Аналоговые и цифровые интегральные схемы [Текст] / С.В. Якубовский. – М.: Советское радио, 1979.